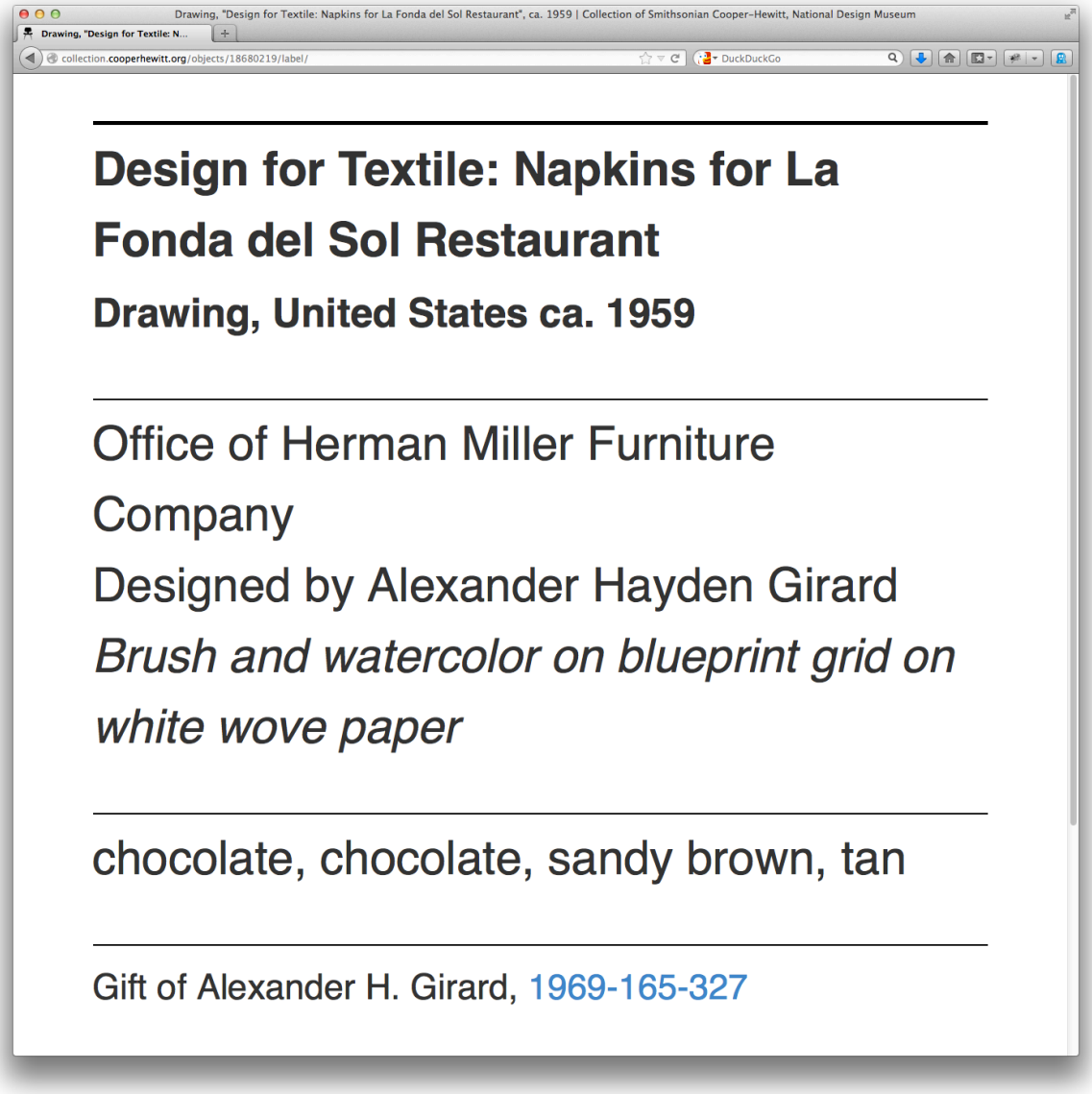
Label Whisperer



http://3ug4ii3uortgp68ee31cybxjyz.wpengine.netdna-cdn.com/wp-content/uploads/2014/01/Screen-Shot-2014-01-24-at-6.06.47-PM.png

Have you ever noticed the way people in museums always take pictures of object labels? On many levels it is the very definition of an exercise in futility. Despite all the good intentions I'm not sure how many people ever look at those photos again. They're often blurry or shot on an angle and even when you can make out the information there aren't a lot of avenues for that data to get back in to the museum when you're not physically in the building. If any-

thing I bet that data gets slowly and painfully typed in to a search engine and then… who knows what happens.

As of this writing the Cooper-Hewitt's luxury and burden is that we are closed for renovations. We don't even have labels for people to take pictures of, right now. As we think through what a museum label should *do* it's worth remembering that cameras and in particular cameras on phones and the software for doing optical character recognition (OCR) have reached a kind of maturity where they are both fast and cheap and simple. They have, in effect, showed up at the party so it seems a bit rude not to introduce ourselves.

I mentioned that we're still working on the design of our new labels. This means I'm not going to show them to you. It also means that it would be difficult to show you any of the work that follows in this blog post without tangible examples. So, the first thing we did was to add a could-play-a-wall-label-on-TV endpoint to each object on the collection website http://collection.cooperhewitt.org/ . Which is just fancy-talk for "another web page".

Simply append `/label` to any object page and we'll display a rough-and-ready version of what a label might look like and the kind of information it might contain. For example:

http://collection.cooperhewitt.org/objects/18680219/label/ http://collection.cooperhewitt.org/objects/18680219/label/

Now that every object on the collection website http://collection.cooperhewitt.org/ has a virtual label we can write a simple print stylesheet http://labs.cooperhewitt.org/2013/cmd-p/ that allows us to produce a physical prototype which mimics the look and feel and size (once I figure out what's wrong with my CSS) of a finished label in the real world.



http://3ug4ii3uortgp68ee31cybxjyz.wpengine.netdna-cdn.com/wp-content/uploads/2014/01/photo-2.jpg

So far, so good. We have a system in place where we can work quickly to change the design of a "label" and test those changes on a large corpus of sample data (the collection http://collection.cooperhewitt.org/) and a way to generate an analog representation since that's what a wall label is.

Careful readers will note that some of these sample labels contain colour information for the object http://labs.cooperhewitt.org/2013/giv-do/ *. These are just placeholders for now. As much as I would like to launch with this information it probably won't make the cut for the re-opening.*

Do you remember when I mentioned OCR software at the beginning of this blog post? OCR software has been around for years and its quality and cost and ease-of-use have run the gamut. One of those OCR application is Tesseract https://code.google.com/p/tesseract-ocr/ which began life in the labs at Hewlitt-Packard and has since found a home and an open source license at Google.

Tesseract is mostly a big bag of functions and libraries but it comes with a command-line application that you can use to pass it an image whose text you want to extract.

In our example below we also pass an argument called `label`. That's the name of the file that Tesseract will write its output to. It will also add a `.txt` extension to the output file because… computers? These little details are worth suffering because when fed the image above this is what Tesseract produces:

```
$> tesseract label-napkin.jpg label
Tesseract Open Source OCR Engine v3.02.01 with Leptonica
$> cat label.txt
_____________j________
Design for Textile: Napkins for La Fonda del
Sol Restaurant

Drawing, United States ca. 1959


_____________________________________
Office of Herman Miller Furniture Company

Designed by Alexander Hayden Girard

Brush and watercolor on blueprint grid on white wove paper

_____________._.._..._____.___._________________________
chocolate, chocolate, sandy brown, tan

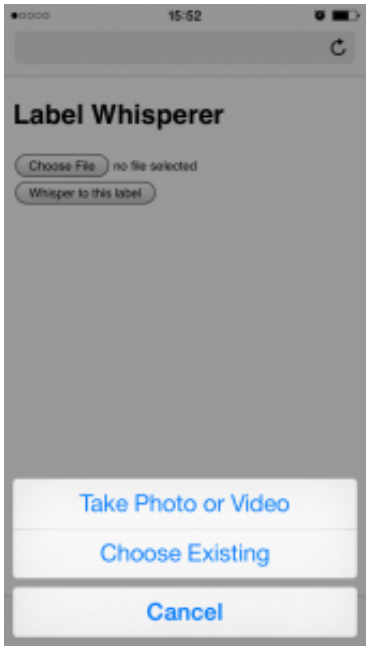

____________________..___._____________________________
Gift of Alexander H. Girard, 1969-165-327
```

I think this is exciting. I think this is exciting because Tesseract does a better than good enough job of parsing and extracting text that I can use that output to look for accession numbers. All the other elements in a wall label are sufficiently ambiguous or unstructured (not to mention potentially garbled by Tesseract's robot eyes) that it's not worth our time to try and derive any meaning from.

Conveniently, accession numbers are so unlike any other element on a wall label as to be almost instantly recognizable. If we can piggy-back on Tesseract to do the hard work of converting pixels in to words then it's pretty easy to write custom code to look at that text and extract things that look like accession numbers. And the thing about an accession number is that it's the identifier *for the thing a person is looking at in the museum*.

To test all of these ideas we built the simplest, dumbest HTTP pony `https://pinboard.in/u:straup/t:httpony` server to receive photo uploads and return any text that Tesseract can extract. We'll talk a little more about the server below but basically it has two endpoints: One for receiving photo uploads and another with a simple form that takes advantage of the fact that on lots of new phones the `file` upload form element on a website will trigger the phone's camera.

This functionality is still early days but is also a pretty big deal. It means that the barrier to developing an idea or testing a theory and the barrier to participation is nothing more than the web browser on a phone. There are lots of reasons why a native application might be better suited or more interesting to a task but the time and effort required to write bespoke applications introduces so much hoop-jumping as to effectively make simple things impossible.

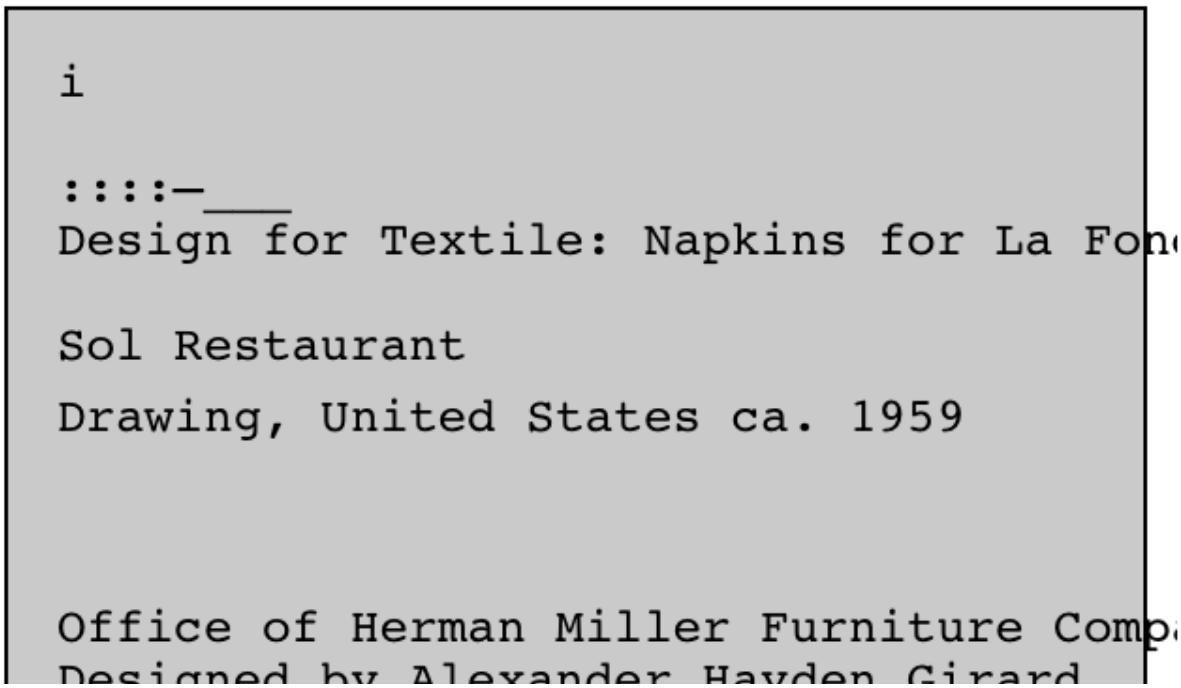


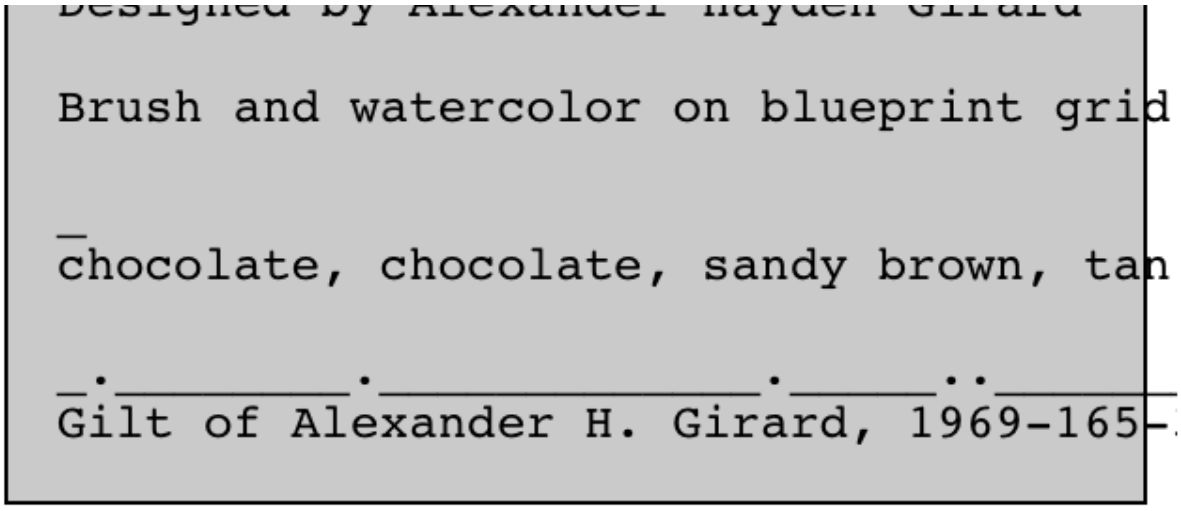
http://3ug4ii3uortgp68ee31cybxjyz.wpengine.netdna-cdn.com/wp-content/uploads/2014/01/photo-2.png

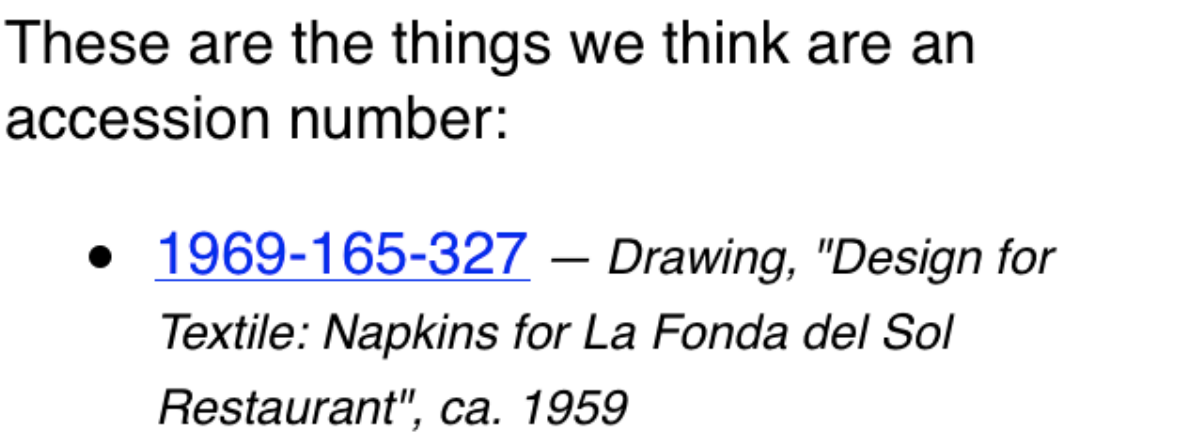


http://3ug4ii3uortgp68ee31cybxjyz.wpengine.netdna-cdn.com/wp-content/uploads/2014/01/photo-3.png

Given a simple upload form which triggers the camera and a submit button which sends the photo to a server we get back pretty much the same thing we saw when we ran Tesseract from the command line:







http://3ug4ii3uortgp68ee31cybxjyz.wpengine.netdna-cdn.com/wp-content/uploads/2014/01/Untitled-cropped.png

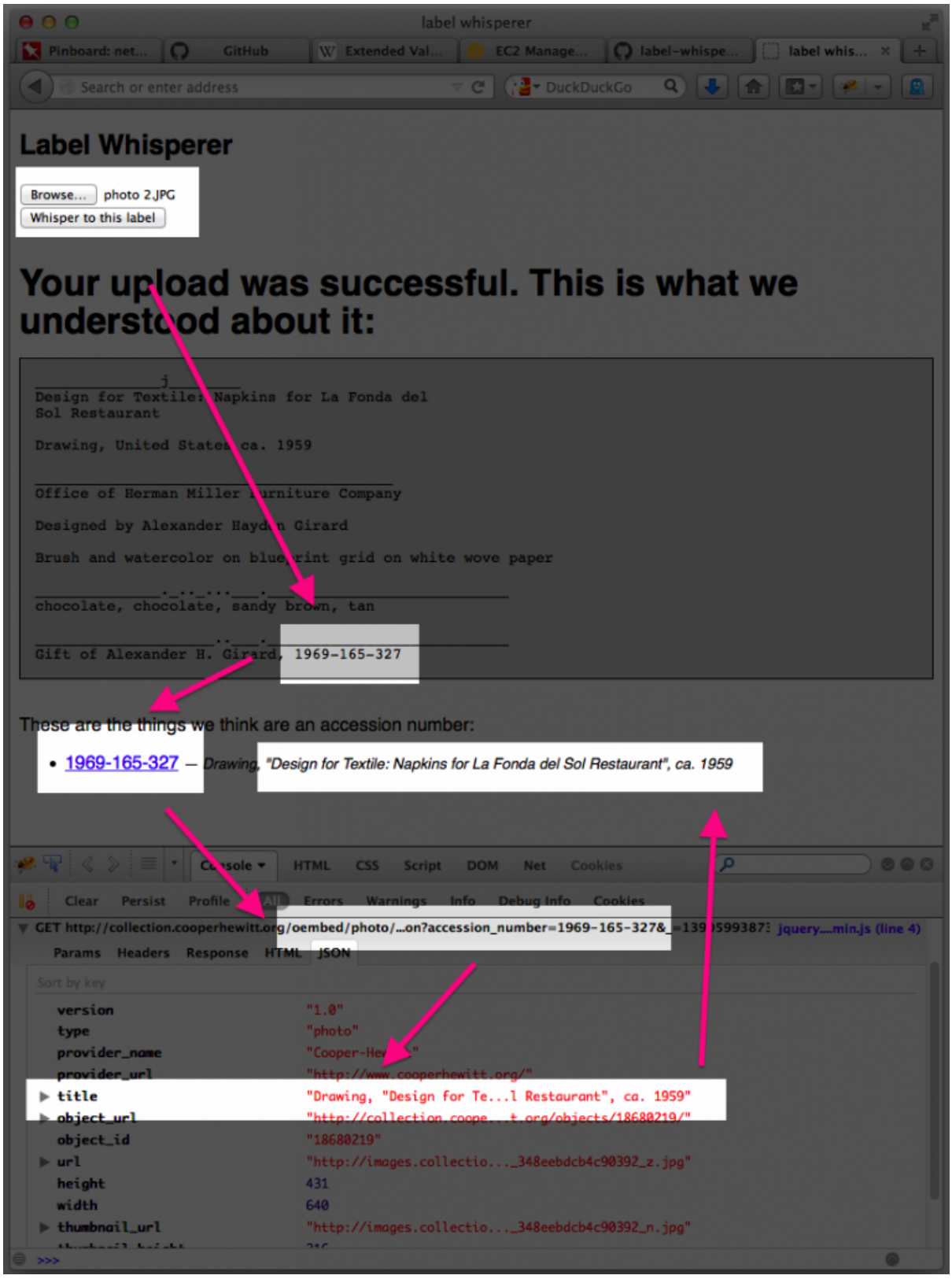
We upload a photo and the server returns the raw text that Tesseract extracts. In addition we do a little bit of work to examine the text for things that look like accession numbers. Everything is returned as a blob of data (JSON) which is left up to the webpage itself to display. When you get down to brass tacks this is really all that's happening:

```
$> curl -X POST -F "file=@label-napkin.jpg" http://localhost | python -mjson.tool
{
    "possible": [
```

```
        "1969-165-327"
    ],
    "raw": "_____j_____nDesign for Textile: Napkins for La Fon-
da delnSol RestaurantnnDrawing, United States ca. 1959n-
n_____nOffice of Herman Miller Furni-
ture CompanynnDesigned by Alexander Hayden GirardnnBrush and watercol-
or on blueprint grid on white wove papern-
n_____._.._..._____.___._____nchocolate, choco-
late, sandy brown, tannn_____.._____._____
nGift of Alexander H. Girard, 1969-165-327"
}
```


Do you notice the way, in the screenshot above, that in addition to displaying the accession number we are also showing the object's title? That information is not being extracted by the "label-whisperer" service. Given the amount of noise produced by Tesseract it doesn't seem worth the effort. Instead we are passing each accession number to the collections website's OEmbed endpoint http://collection.cooperhewitt.org/oembed/ and using the response to display the object title.

Here's a screenshot of the process in a plain old browser window with all the relevant bits, including the background calls across the network where the robots are talking to one another, highlighted.



http://3ug4ii3uortgp68ee31cybxjyz.wpengine.netdna-cdn.com/wp-content/uploads/2014/01/label-whisperer-napkin-boxes.png

1. Upload a photo
2. Extract the text in the photo and look for accession numbers
3. Display the accession number with a link to the object on the CH collection website http://collection.cooperhewitt.org/
4. Use the extracted accession number to call the CH OEmbed endpoint http://collection.cooperhewitt.org/oembed for additional information about the object
5. Grab the object title from the (OEmbed) response and update the page

See the way the OEmbed response contains a link to an image for the object? See the way we're not doing anything with that information? Yeah, that…

But we proved that it can be done and, start to finish, we proved it inside of a day.

It is brutally ugly and there are still many failure states but we can demonstrate that it's possible to transit from an analog wall label to its digital representation on a person's phone. Whether they simply bookmark that object or email it to a friend or fall in to the rabbit hole of life-long scholarly learning is left an as exercise to the reader. That is not for us to decide. Rather we have tangible evidence that there are ways for a museum to adapt to a world in which all of our visitors have super-powers — aka their "phones http://craigmod.com/journal/photography_hello/" — and to apply those lessons to the way we design the museum itself.

We have released all the code and documentation required build your own "label whisperer" under a BSD license but please understand that it is only a *reference implementation*, at best. A variation of the little Flask http://flask.pocoo.org/ server we built might eventually be deployed to production but it is unlikely to ever be a public-facing thing as it is currently written.

https://github.com/cooperhewitt/label-whisperer/ https://github.com/cooperhewitt/label-whisperer/

We welcome any suggestions for improvements or fixes that you might have. One important thing to note is that while accession numbers are pretty straightforward there are variations and the code as it written today *does not account for them*. If nothing else we hope that by releasing the source code we can use it as a place to capture and preserve a catalog of patterns because life is too short to spend very much of it training robot eyes to recognize accession numbers.

The whole thing can be built without any external dependencies if you're using Ubuntu 13.10

and if you're not concerned with performance can be run off a single "micro" Amazon EC2 instance. The source code contains a handy setup script https://github.com/cooperhewitt/label-whisperer/blob/master/ec2/setup.sh for installing all the required packages.

Immediate next steps for the project are to make the `label-whisperer` server hold hands with Micah's Object Phone http://labs.cooperhewitt.org/2013/object-phone/ since being able to upload a photo as a text message would make all of this accessible to people with older phones and, old phone or new, requires users to press fewer buttons. Ongoing next steps are best described as "learning from and doing everything" talked about in the links below:

- Michal Migurski's Walking Papers http://mike.teczno.com/notes/walking-papers.html and Walking Papers Cheaply http://mike.teczno.com/notes/walking-papers-cheaply.html
- Astronomy.net's Making the Sky Searchable http://cosmo.nyu.edu/hogg/research/2006/09/28/astrometry_google.pdf
- The Royal Observatory's Introducing Astrotags http://vimeo.com/6469344 — *if you don't bother following any of the other links at least watch this because it's basically the best thing ever* http://vimeo.com/4717981
- Matt Jones' Product Sketch: Clocks for Robots http://berglondon.com/blog/2011/09/22/clocks-for-robots/

Discuss!

This entry was posted in CH 3.0 http://labs.cooperhewitt.org/category/ch-3-0/, *Experimental* http://labs.cooperhewitt.org/category/experimental/, *Papernet* http://labs.cooperhewitt.org/category/papernet/, *Publishing* http://labs.cooperhewitt.org/category/publishing/ *on January 25, 2014* [http://labs.cooperhewitt.org/2014/label-whisperer/] *by asc* http://labs.cooperhewitt.org/author/asc/.

Dataclimber explores colors in the Cooper Hewitt collection



Rubén Abad's #museumselfie outside of a museum

A few weeks ago we became aware of Rubén Abad's poster http://dataclimber.net/blog/2014/1/19/cooper-hewitts-collection-color-history which shows all the colours in our collection by decade. We sent a few questions over to Spain to find out more . . .

Q: What were some of the precursors to the color poster? What inspired you?

A: The idea came when I first saw Lev Manovich's 'Software Takes Command http://www.manovich.net/softbook/ ' book cover. When I started looking at the data, another couple of paintings came to my mind. For example, Salvador Dalí's series about visual perception and 'pixels', as in Homage to Rothko http://thedali.org/exhibits/highlights/gala_contemplating_the_mediterranean_sea_which_at_twenty_meters_becomes_the_portrait_of_abraham_lincoln.php (The Dalí Museum). By chance, I attended an exhibition here in Madrid where I discovered 'Study for Index: Map of the World http://www.macba.cat/en/study-for-index-map-of-the-world-al0166 ', by Art & Language (MACBA). By the time I came back home, it was clear that I wanted to display color evolution over time using a mosaic.

Q: Did you have any expectation about what the final product would look like? Did the

end result surprise you?

A: I didn't have any preconceived notion. I liked to see how groups of pieces appeared.

Q: What were the challenges of working with the dataset? What were the holes, problems? How could we make it better/easier to work with?

A: Being used to work with data made really easy for me to work with the collection's dataset, so thanks for releasing it! The only complain I might have is having to parse some fields, like medium, to be able to store the information in a more comfortable format to be queried.

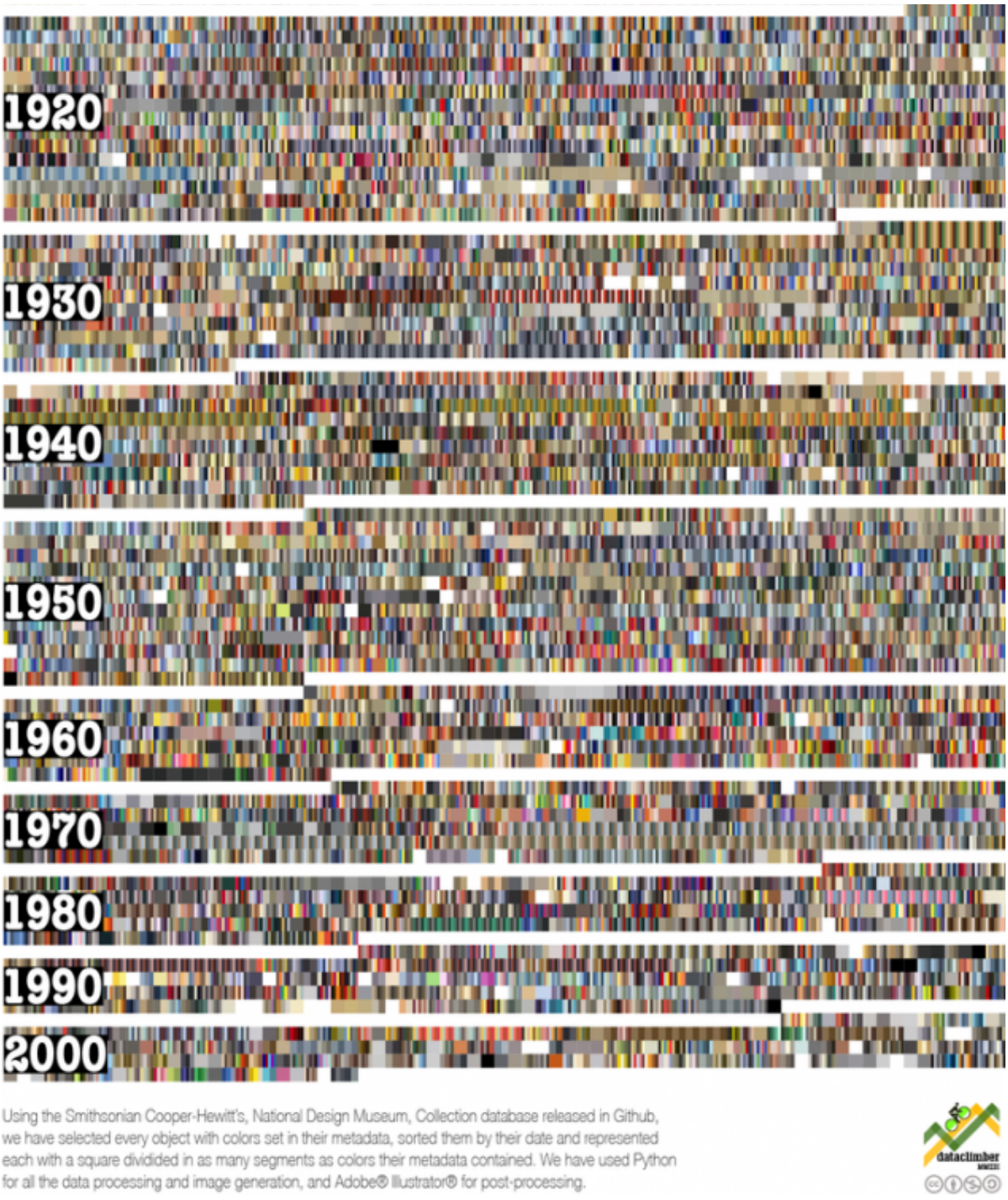
Q: What would you like to do next?

A: I have a network of people and objects in mind, in order to display who has the biggest 'influence' in the collection.

Q: If other museums made their data available like this, what might you do with it?

A: I'd like to work on a history of the object project. If we were able to access all the dates and places importants in the object history, we could try to cross all the objects info and maybe, it's never known, find new hubs where pieces happened to be at the same time and why they were there. Another interesting project would be to find gender inequality among collections, not only when looking at artists/designers, but also with donors and funders and even among representations (iconography). Have this roles changed over the years? Are different depending on countries?





http://dataclimber.net/blog/2014/1/19/cooper-hewitts-collection-color-history

Dataclimber's color poster.

This entry was posted in Collection data http://labs.cooperhewitt.org/category/collection-data/ *on February 2, 2014* [http://labs.cooperhewitt.org/2014/dataclimber-explores-colors-in-the-cooper-hewitt-collection/] *by Seb*

Chan http://labs.cooperhewitt.org/author/seb/.

Video Capture for Collection Objects

Stepping inside a museum storage facility is a cool experience. Your usual gallery ambience (dramatic lighting, luxurious swaths of empty space, tidy labels that confidently explain all) is completely reversed. Fluorescent lights are overhead, keycode entry pads protect every door, and official ID badges are worn by every person you see. It's like a hospital, but instead of patients there are 17th century nightgowns and Art Deco candelabras. Nestled into tiny, sterile beds of acid-free tissue paper and archival linen, the patients are occasionally woken and gently wheeled around for a state-of-the-art microscope scan, an elaborate chemical test, or a loving set of sutures.



http://3ug4ii3uortgp68ee31cybxjyz.wpengine.netdna-cdn.com/wp-content/uploads/2014/02/Screen-Shot-2014-02-13-at-3.18.09-PM.png

A rare peek inside the storage facility.

If you ask a staff member for an explanation of this or that object on the nearest cart or shelf, they might tell you a detailed story, or they might say that so far, not much is known. I like the element of unevenness in our knowledge, it's very different from the uniform level of confidence one sees in a typical exhibition.

The web makes it possible to open this space to the public in all its unpolished glory – and

many other museums have made significant inroads into new audiences by pulling back the curtain. The prospect is like catnip for the intellectually curious, but hemlock for most museum employees.

Typically, the only form of media that escapes this secretive storage facility are hi-res TIFFs artfully shot in an on-site photography studio. The seamless white backdrop and perfectly staged lighting, while beautiful and ideal for documentation, completely belie the working lab environment in which they were made.

We just launched a new video project called “Collections in Motion.” The idea is super simple: short videos that demonstrate collections objects that move, flip, click, fold, or have any moveable part.

Here are some of the underlying thoughts framing the project:

- Still images don’t suffice for some objects. Many of them have moving parts, make sounds, have a sense of weight, etc that can’t be conveyed through images.
- Our museum’s most popular videos on YouTube are all kinetic, kinda entrancing, moving objects. (Contour Craft 3D Printing https://www.youtube.com/watch?v=-yv-IWdSdns , A Folding Bicycle https://www.youtube.com/watch?v=H7alqIkPKsg , and a Pop-up Book https://www.youtube.com/watch?v=mVvj_QFtWqk , for example).
- Videos played in the gallery generally don’t have sound or speakers available.
- In research interviews with various types of visitors, many people said that they wouldn’t be interested in watching a long, involved video in a museum context.
- Animated GIFs, 6-second Vines, and 15-second Instagram videos loom large in our contemporary visual/communication culture.
- How might we think of the media we produce (videos, images, etc) as a part of an iterative process that we can learn from over time? Can we get comfortable with a lower quality but higher number of videos going out to the public, and seeing what sticks (through likes, comments, viewcount, etc)?

Top 10 Videos Browse all content

Video	Views ↓	Estimated minutes watched	Likes
Why Design Now?: Contour Crafting	333,431	862,783	631
Why Design Now?: IF Mode Folding Bicycle	11,702	20,378	30
Cooper-Hewitt - Pop-up book: Puss in Boots	3,241	3,562	4
Why Design Now?: Vault201	3,060	3,296	1
Why Design Now?: Loblolly House	2,666	11,299	4

http://3ug4ii3uortgp68ee31cybxjyz.wpengine.netdna-cdn.com/wp-content/uploads/2014/02/Screen-Shot-2014-02-13-at-12.36.32-PM.png

Our most popular YouTube videos for this quarter. They are all somewhat mesmerizing/cabinet-of-curiosity type things.

Here are some of the constraints on the project:

- No budget (pairs nicely with the preceding bullet).
- Moving collections objects is a conservation no-no. Every human touch, vibration and rub is bad for the long-long-longevity of the object (and not to mention the peace of mind of our conservators).
- Conservators' and curators' time is in HIGH demand, especially as we get closer to our re-opening. They are busy writing new books, crafting wall labels, preparing gallery displays, etc. Finding a few hours to pull an object from storage and move it around on camera is a big challenge.

So, nerd world, what do you think?

This entry was posted in Collection data http://labs.cooperhewitt.org/category/collection-data/, *Digitization* http://labs.cooperhewitt.org/category/digitization/, *Experimental* http://labs.cooperhewitt.org/category/experimental/ *on February 14, 2014* [http://labs.cooperhewitt.org/2014/collections-in-motion/] *by katieshelly* http://labs.cooperhewitt.org/author/katieshelly/.

Downgrading your website (or why we are moving to WordPress)

Below are the slides and most of what I said at the 2014 Museums & The Web http://www.museum-sandtheweb.com/ conference in Baltimore, Maryland.

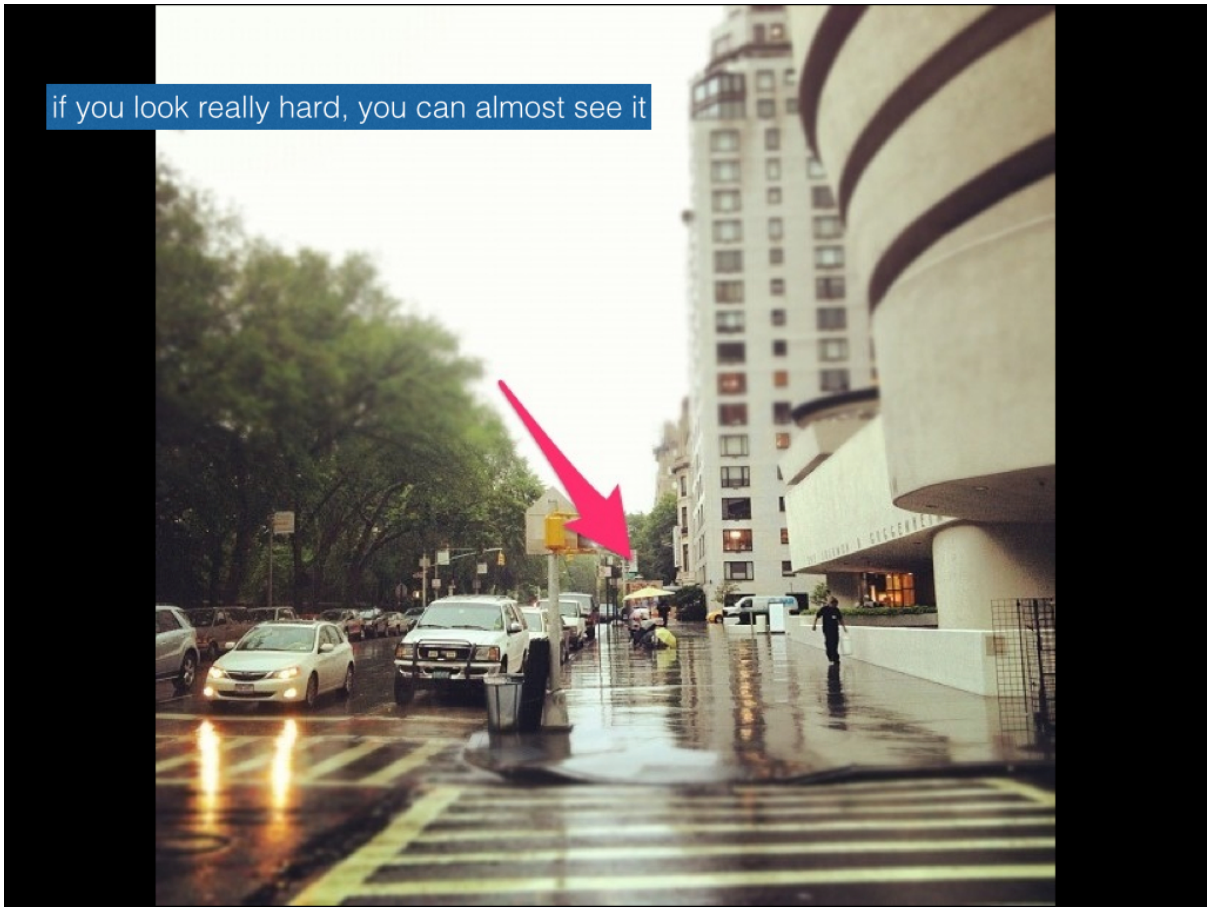
> *"I believe that if we think first about people and then try, try, and try again to prototype our designs, we stand a good chance of creating innovative solutions that people will value and enjoy."—Bill Moggridge*



http://3ug4ii3uortgp68ee31cybxjyz.wpengine.netdna-cdn.com/wp-content/uploads/2014/04/MW2014.002.jpg

Let me begin by telling you a little story about a small museum that sat along 5th Ave. on

New York's Upper East Side. This is of course a largely fictional story. Names, and actual events have been changed.



http://3ug4ii3uortgp68ee31cybxjyz.wpengine.netdna-cdn.com/wp-content/uploads/2014/04/MW2014.003.jpg

This is the story of a little museum with big aspirations. Long ago this little museum had a website. It had a webmaster, and it published a blog. It even had a whole bunch of microsites, flash driven exhibition sites, event calendars and archives. In fact, it won a few Webby's.



http://3ug4ii3uortgp68ee31cybxjyz.wpengine.netdna-cdn.com/wp-content/uploads/2014/04/MW2014.004.jpg

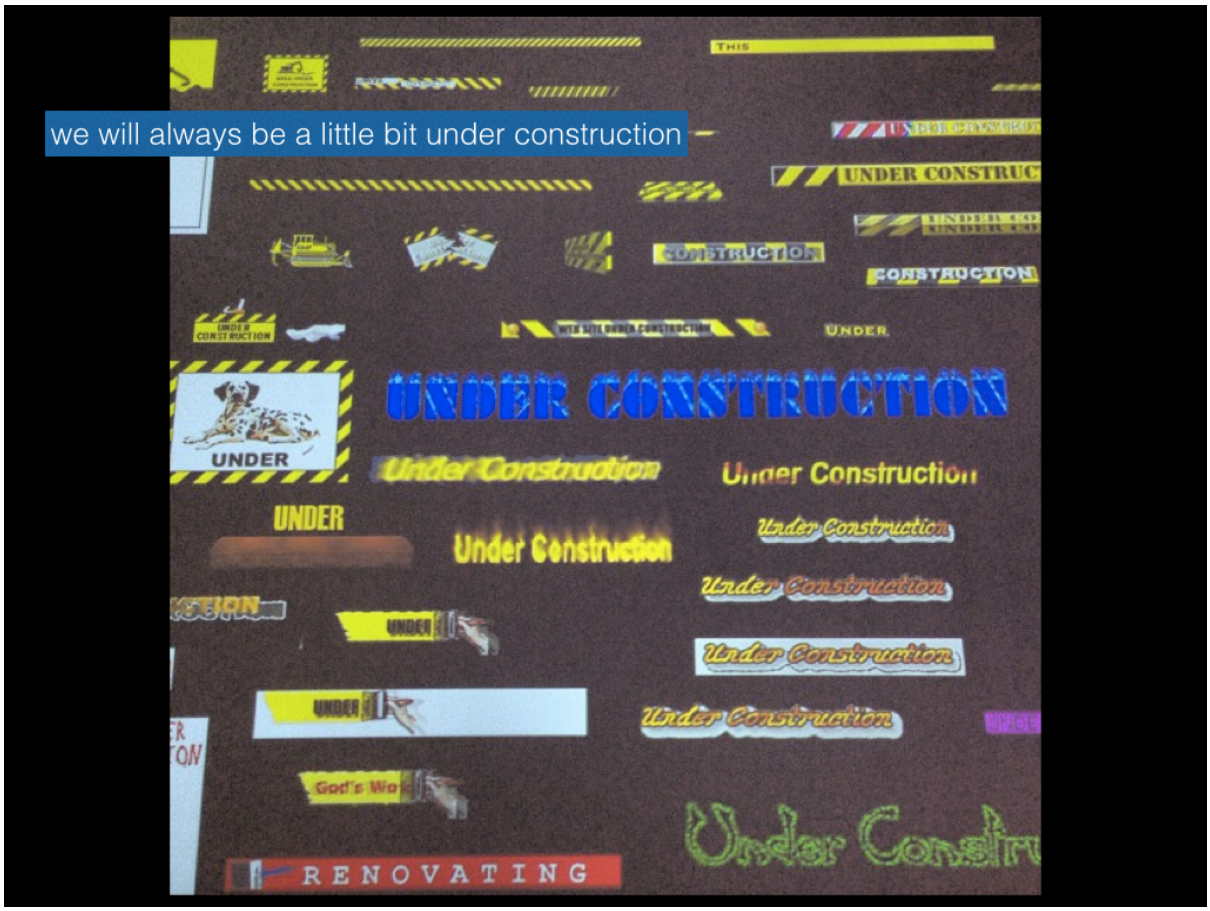
The website was very much the product of an organization trying to get the job done. And, it succeeded in this effort. Staff members would produce content on their company issued PCs and would then hand these documents off to the museum's webmaster who would convert them into HTML and Javascript. The webmaster would press a specially designed "button" which would upload the new content to the little museum's web servers where the pages would be served and maintained by a giant umbrella organization that had close ties to the government.



http://3ug4ii3uortgp68ee31cybxjyz.wpengine.netdna-cdn.com/wp-content/uploads/2014/04/MW2014.005.jpg

With a single webmaster managing the entirety of the museum's web properties, the little staff of this museum faced an inevitability. It was just too much work for the webmaster to do alone. Even if they allowed the webmaster an apprentice, the workload would continue to grow, and the little museum's website would suffer. Eventually, they all realized they would have to move towards a system that would allow the entire staff to collaborate more efficiently.

Eventually, they realized they would need a content management system.



http://3ug4ii3uortgp68ee31cybxjyz.wpengine.netdna-cdn.com/wp-content/uploads/2014/04/MW2014.006.jpg

There were many options out there already, and the little museum's webmaster took stock in as many of them as he could. Meetings were had, and budgets were considered. The "committee to select a content management system" was formed, and consultants were brought in.

Wire frames were presented, and scopes of work were proposed, but the committee remained vigilant and put off making a decision as long as it could. They simply never felt like they had the right solution placed in front of them.

There was a lot at stake and many facets and bullet points drove them to a moment of indecision. There was due-dilligence due to their "mothership" in Washington, and there were "rights in data" clauses to be haggled over, with threats of time in a Federal prison always on everyone's minds. Eventually the committee was disbanded and the project was put on hold.



http://3ug4ii3uortgp68ee31cybxjyz.wpengine.netdna-cdn.com/wp-content/uploads/2014/04/MW2014.007.jpg

Time went on and the little museum's website continued to shine as the public face of the institution. It continued to be updated with more and more content, and eventually the little museum even invested a fair amount of money in putting their collections online for all to see.

The word on the street was that this little museum's website was starting to blow up, more and more people were beginning to rely on it as source of good information, and the time had come to re-think the idea of re-building.



http://3ug4ii3uortgp68ee31cybxjyz.wpengine.netdna-cdn.com/wp-content/uploads/2014/04/MW2014.008.jpg

The webmaster at the little museum was doing his best, running around from staff member to staff member, trying to understand what had been going on all this time. One day he had the fortune to sit in on a meeting with a prominent weblogger and asked him a very important question.

“What CMS do you think we should chose” the webmaster said.

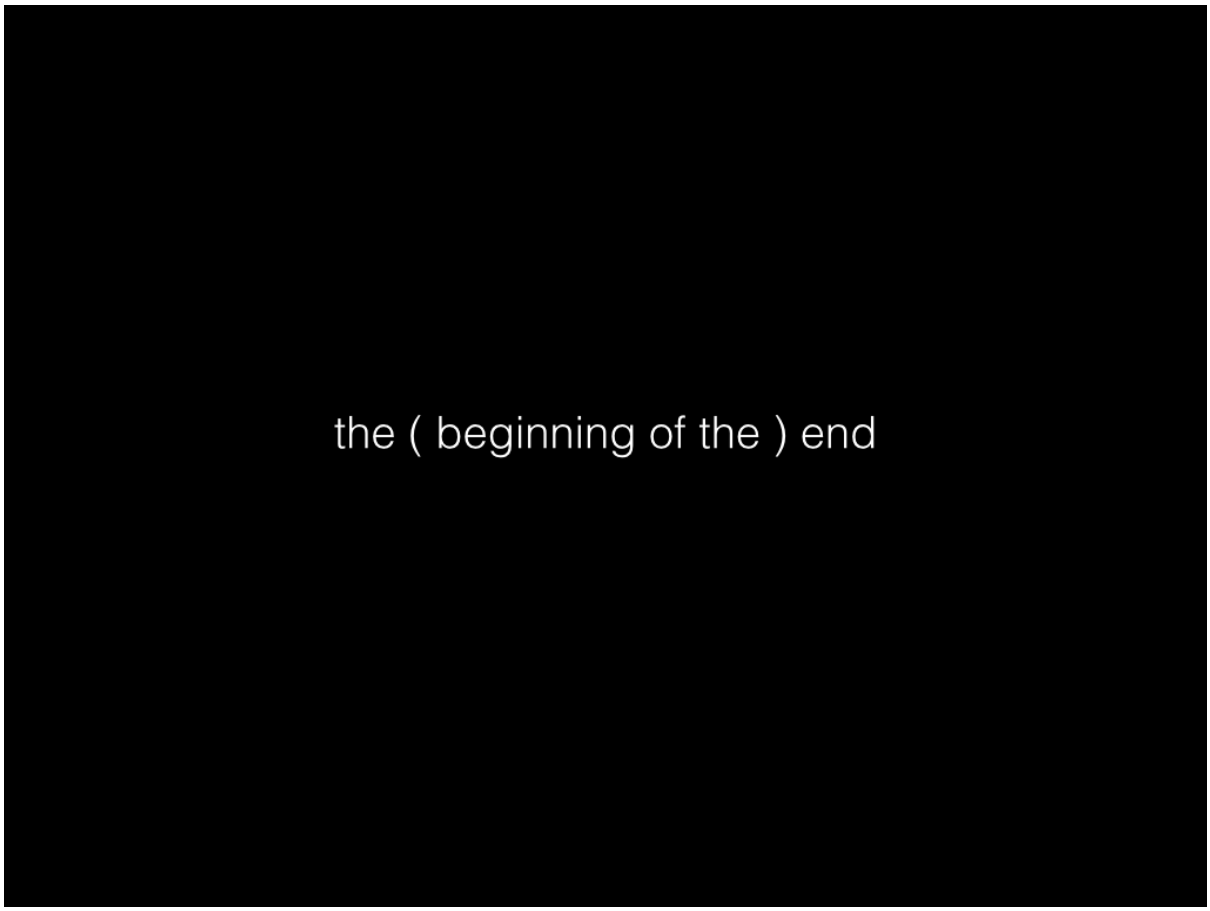
“CMS’s are all basically the same”, said the blogger, “just chose one you like and don’t look back.”

The webmaster took this to heart and selected three CMS systems that were free and easy to set up. He presented these to the higher ups and after a couple of hours of debate and one technical review board meeting, the webmaster had his answer.



http://3ug4ii3uortgp68ee31cybxjyz.wpengine.netdna-cdn.com/wp-content/uploads/2014/04/MW2014.009.jpg

Drupal would be the content management system for the little museum. Drupal.



http://3ug4ii3uortgp68ee31cybxjyz.wpengine.netdna-cdn.com/wp-content/uploads/2014/04/MW2014.010.jpg

The end, well sorta.

Most of that actually happened at the Cooper-Hewitt. The team eventually just had to pick a system, (without a whole lot of experience with the product itself) and kind of just “go for it.” From that point on, the staff at Cooper-Hewitt were living with Drupal. Drupal, a word almost none of the staff had ever heard before became, in less than a few months, a dirty word, spoken in fits of anger and dismay.

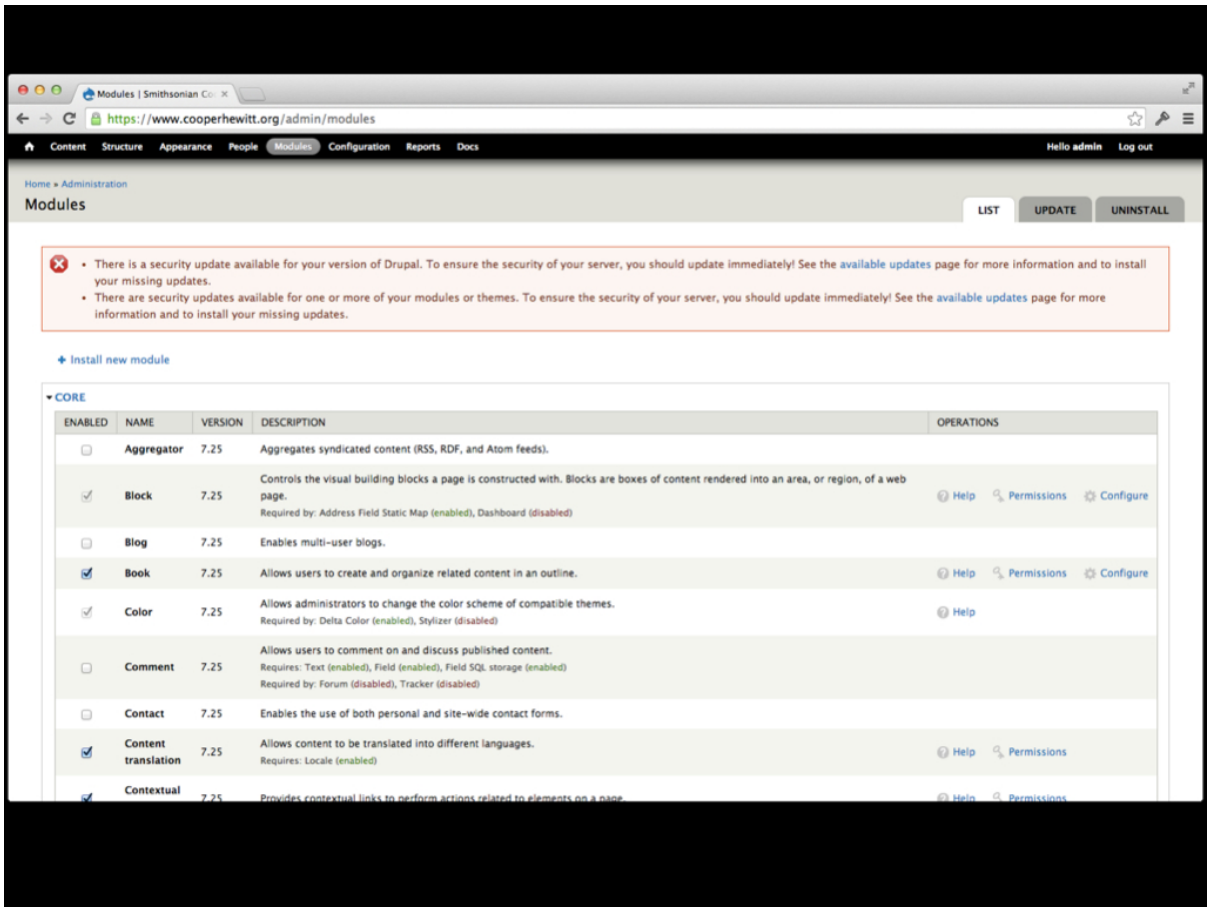
Now, before we go any further, it really needs to be said out loud that Drupal is really fine piece of software that has grown and evolved into a very sophisticated and well thought out framework for building websites. It has a rich community of developers and enthusiasts behind it and it powers some of the most popular websites on the planet. It’s used by giant companies far and wide, governments, and educational institutions all over. As well, our team in Washington has come a really long way in learning how to host and maintain Drupal based websites and presently, many of the latest Smithsonian websites are being built on Drupal. There is nothing intrinsically wrong with Drupal, we just realized, after a long time, it

wasn't for us.



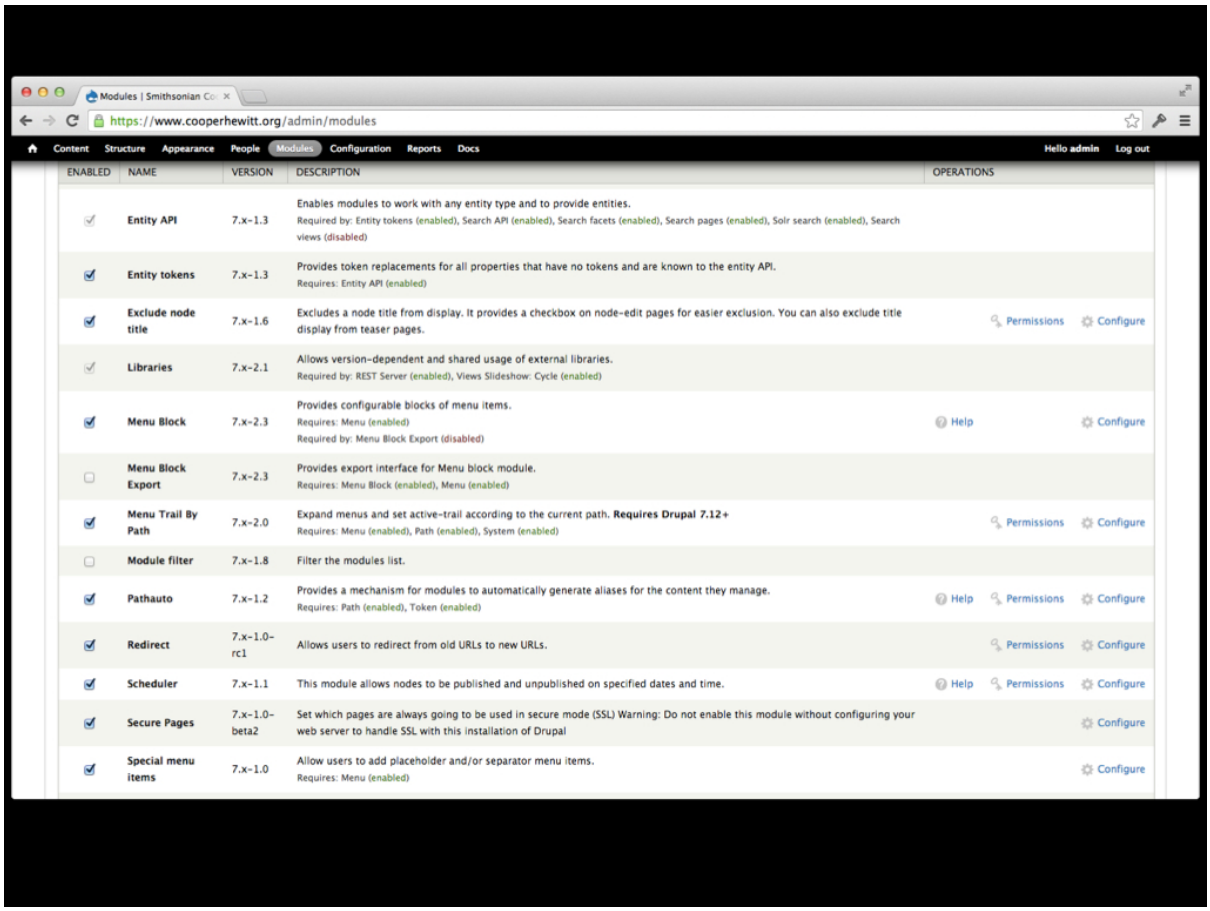
http://3ug4ii3uortgp68ee31cybxjyz.wpengine.netdna-cdn.com/wp-content/uploads/2014/04/MW2014.011.jpg

I'm Micah Walter. I'm part of the nerd crew at Cooper-Hewitt. We are part of the Smithsonian (that umbrella corporation in Washington)… and we are in the middle of a re-launch of our physical museum, as well as our digital presence.



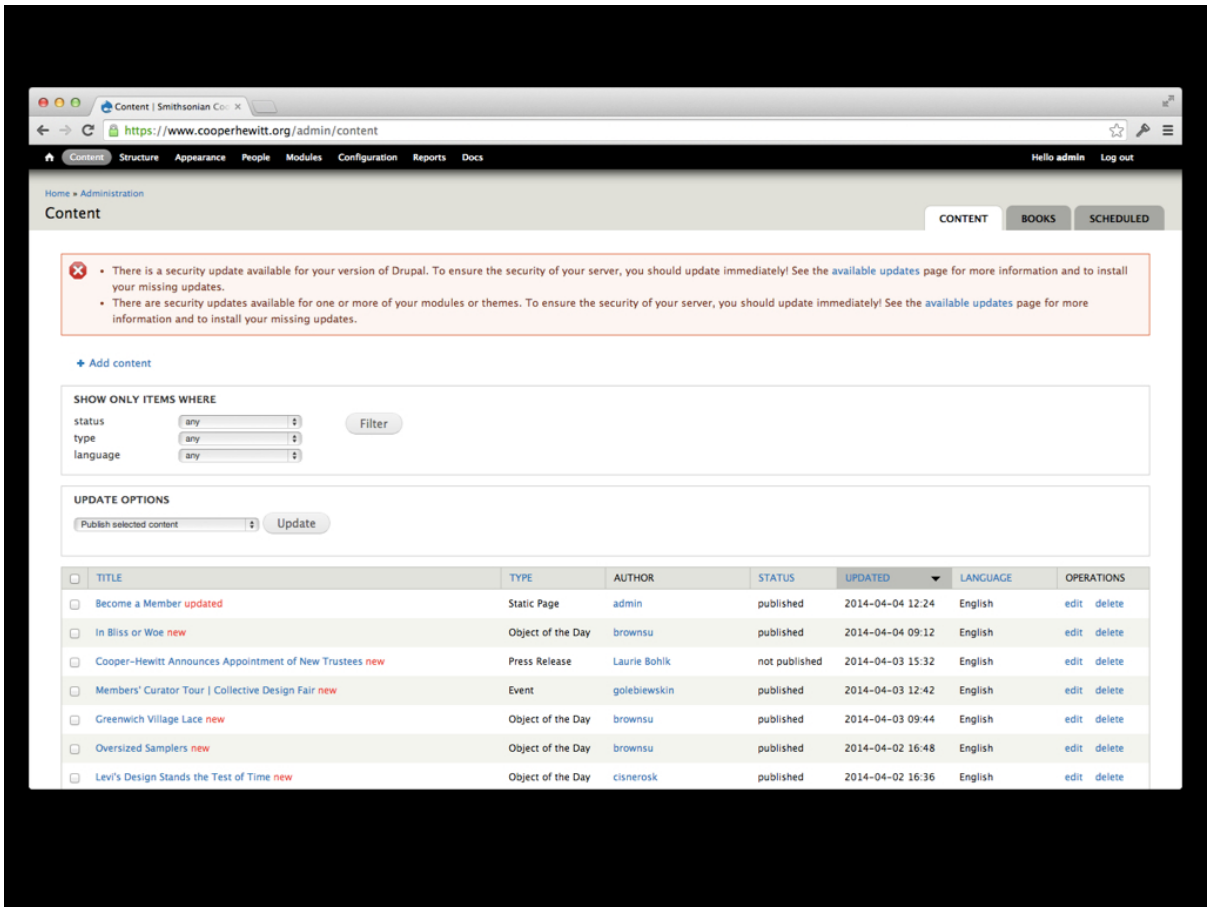
http://3ug4ii3uortgp68ee31cybxjyz.wpengine.netdna-cdn.com/wp-content/uploads/2014/04/MW2014.012.jpg

Cooper-Hewitt started it's life with a CMS by installing a copy of Drupal 6. Shortly thereafter, we installed some modules, and more modules, and more…modules. Eventually we had a pretty awesome website. We hired an engineering team to convert the look and feel of the old website into a Drupal theme, and we "went live." Cooper-Hewitt was on a CMS and it felt good.



http://3ug4ii3uortgp68ee31cybxjyz.wpengine.netdna-cdn.com/wp-content/uploads/2014/04/MW2014.013.jpg

random extra slide

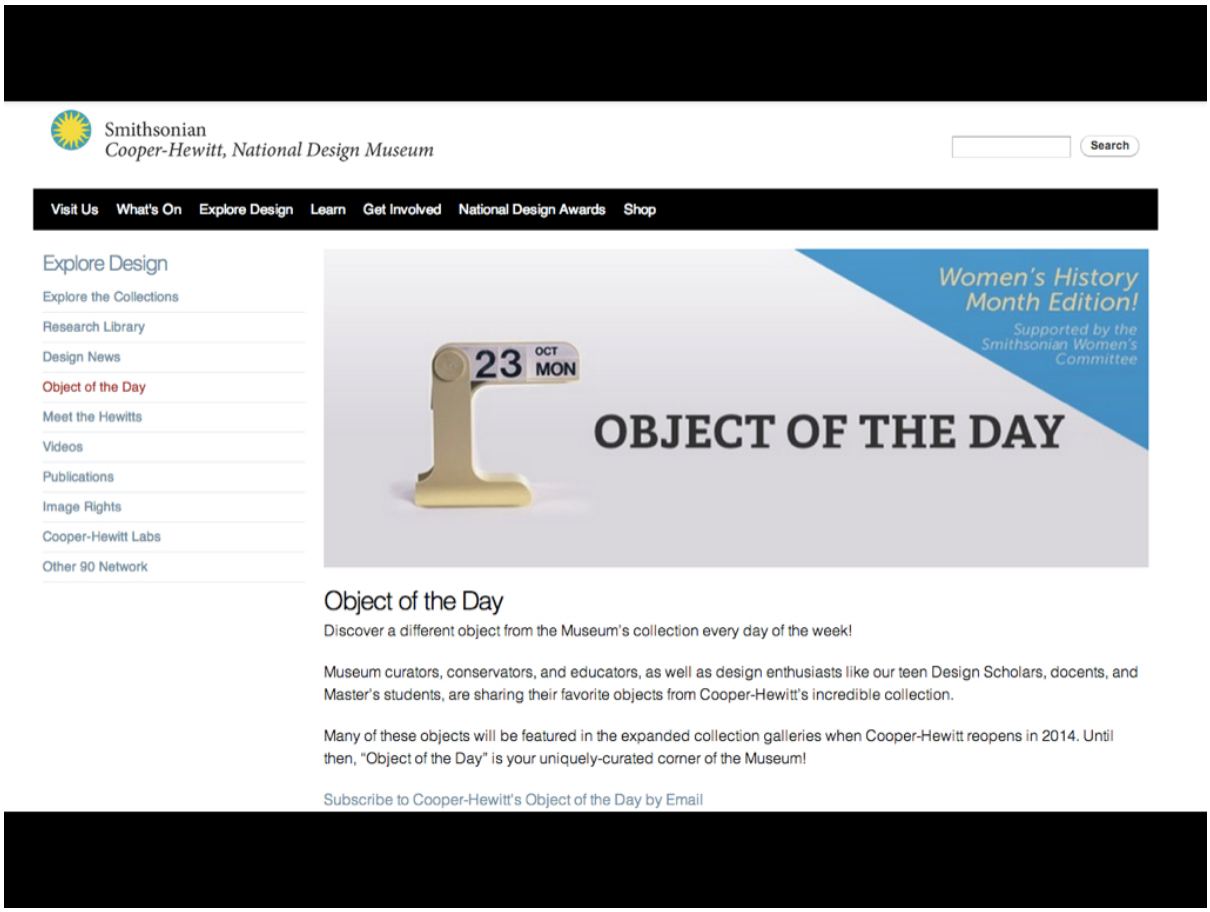


http://3ug4ii3uortgp68ee31cybxjyz.wpengine.netdna-cdn.com/wp-content/uploads/2014/04/MW2014.014.jpg

Some time during this process we sat down with all the staff members to show off our new CMS. We took them on a tour of the system and poked around with a few of the CMSs features, with the hopes of getting staffers excited about the whole thing. The staff seemed to respond positively, and after a couple of months of configuring Drupal's permissions matrix, we gave out login details to a select number of "power users" around the museum. A few of these power users got it right away and were off and running, updating their existing webpages when they needed to. It wasn't too bad actually. Staff could easily log in, search around for the relevant content and make minor changes to their pages. The problems started to appear when they wanted to do just slightly more. A staffer wasn't able to easily upload an image to Drupal. The image had to first be sent to our graphics person who would convert it to a jpeg, resize it for the web and then it would be sent to the webmaster who would upload it to an Amazon S3 server. Once this was done the webmaster would email the URL to the image back to the staffer who would then try and figure out how to insert it into their page.

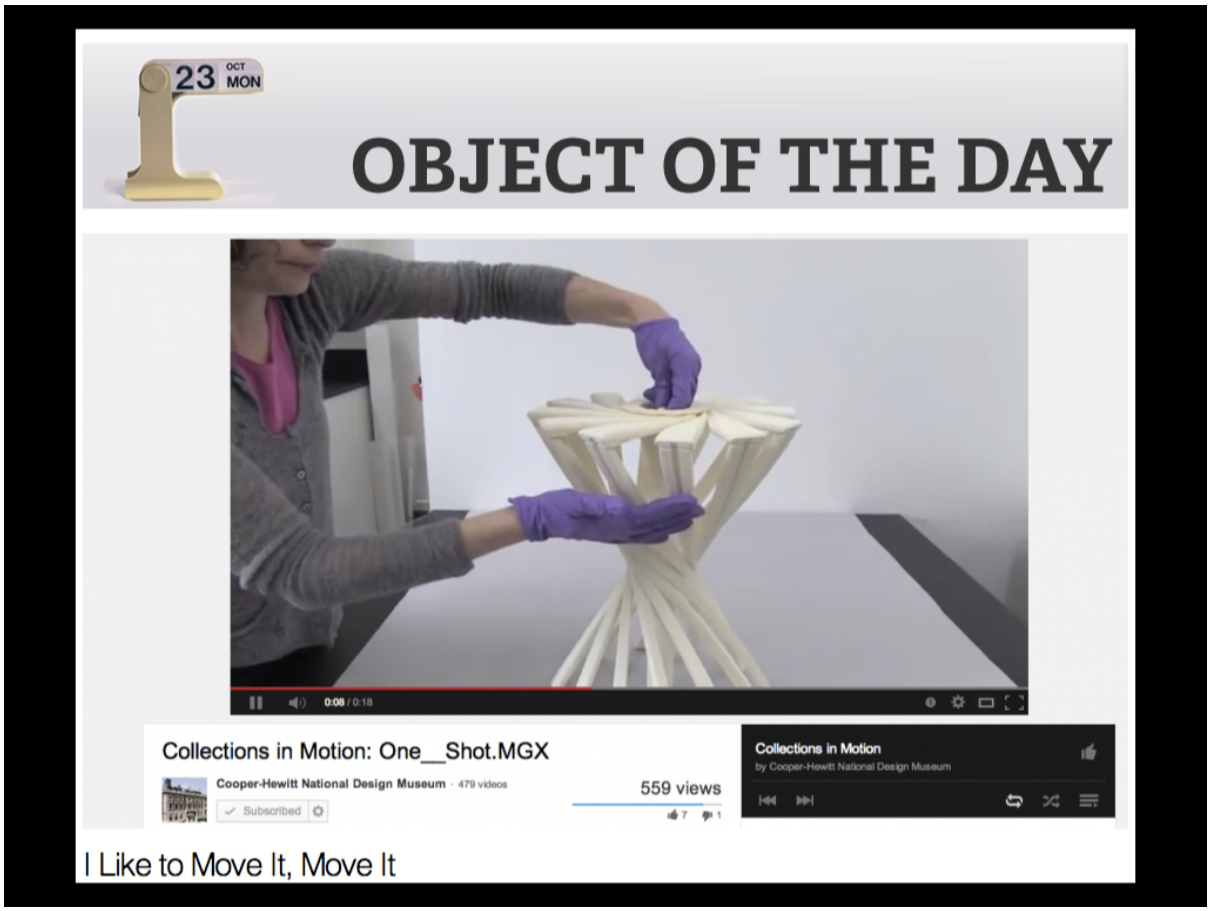
Another issue arose when staffers tried to author new pages. It was simply difficult for them to understand how the new page would find its way within the information architecture that

was already in place. How were they to set the new page's URL and menu items. Those kinds of tasks inevitably wound up back on the webmaster's desk.



http://3ug4ii3uortgp68ee31cybxjyz.wpengine.netdna-cdn.com/wp-content/uploads/2014/04/MW2014.015.jpg

For the most part, notwithstanding a few hiccups here and there, Drupal 6 ran pretty smoothly. Staffers were able to distribute the workload a little more than they used to, and that was considered a good thing. But, about a year into it, a grant became available and the notion of running a daily blog about our objects turned into a reality. Object of the Day was born, and we had our work cut out for us.



http://3ug4ii3uortgp68ee31cybxjyz.wpengine.netdna-cdn.com/wp-content/uploads/2014/04/MW2014.016.jpg

Object of the Day went through many stages of evolution, eventually winding up as an institutional blog authored by staffers, students in our Masters program, docents, and even teens and high school kids interested in design. Every day another object from the collection was chosen and a post was written about it and published to our blog. Great pains were taken to ensure we considered the collection record, tags, the authors vitals and more. We met in committee meetings over and over and eventually worked out a plan to allow us to manage project. The end result would be a new post about a different object, every day.

In the beginning we toyed around with the idea of Object of the Day being run on a separate platform. We considered Tumblr, WordPress.com and even Blogger. But in the end, we decided we would put our new CMS to the test and put ourselves through the process of managing a daily blog with Drupal.



http://3ug4ii3uortgp68ee31cybxjyz.wpengine.netdna-cdn.com/wp-content/uploads/2014/04/MW2014.017.jpg

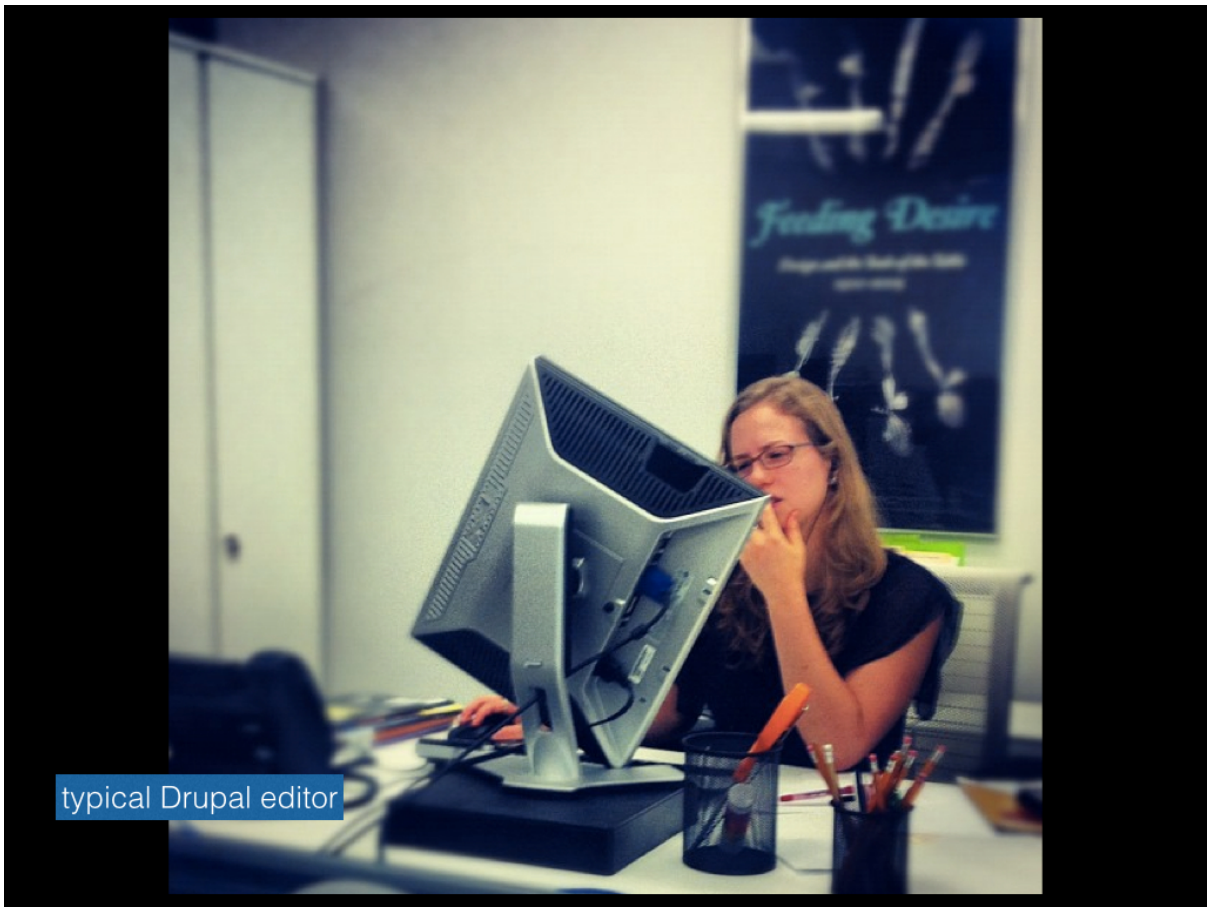
To accomplish this, the digital team realized we'd probably be wise to migrate to Drupal 7 in order to take advantage of its much improved back end user interface. So, with Object of the Day as catalyst, we moved ahead with plans to migrate our Drupal installation to D7. Consultants were hired, interns were enslaved and the whole process took just a few months. In the end we wound up with a fresh installation of Drupal 7, and about 20 or so contributed modules.



http://3ug4ii3uortgp68ee31cybxjyz.wpengine.netdna-cdn.com/wp-content/uploads/2014/04/MW2014.018.jpg

In parallel to this migration project we began to meet with staff members and work out the details of how this Object of the Day project would go down. We discussed a variety of organizational schemes, we talked about available resources, and how far the grant money might take us. In the end we came up with a pretty simple plan. Each month, one staff member would be the “editor” for Object of the Day. He or She would be responsible for collecting all the entries for the month, making sure they were entered into Drupal, edited and fact checked. They would then get scheduled to be published automatically on their specific day. This included many spreadsheets, checklists and meetings. It was of course, great user research for me and my team.

Once we had D7 up and running staff members started to get the hang of it. They started logging in and authoring content. And then the problems started to happen.



http://3ug4ii3uortgp68ee31cybxjyz.wpengine.netdna-cdn.com/wp-content/uploads/2014/04/MW2014.019.jpg

We already had about 1500 pages (Drupal calls these nodes) in the CMS. They were mostly static web pages about one program or another, or blog posts from the old days, or exhibition archives and other kinds of historic content. This was just fine as that content rarely got touched or updated. It was also fine when we wanted to add a fresh blog post or a new static page every once in a while…

The problem though was what happened when the monthly Object of the Day editor had to log in to start work on their thirty some posts for the upcoming month. It was nearly impossible for them to collect all the posts in one place within the CMS so that they could see what had been entered, what was finalized and what was ultimately scheduled. This was a major first hiccup and the digital team worked out a solution involving a number of custom Drupal views that would allow the editors to more easily see what they were working on. It kind of worked, but we could tell that it was a hack solution to a real problem.

The end result was, they lived with it. They lived with the system, learned to hate it, and just didn't talk about it much. Drupal became this beast that they just came to terms with.



http://3ug4ii3uortgp68ee31cybxjyz.wpengine.netdna-cdn.com/wp-content/uploads/2014/04/MW2014.020.jpg

Time went one, and we all learned to work with Drupal. Many of the staff members became proficient enough to get by, and the calls to the webmaster desk lessened. But, the problems hadn't gone away. In fact our little experiment to try and get staff members excited about authoring content on the web had actually backfired. Now, staff members authored content for Object of the Day because it was part of their job, listed in their work plans and reviewed during their performance evaluations at the end of each year. They hated it.

Meanwhile, Object of the Day took off. The public facing version of the blog became a big success. It received additional funding for a second year with the idea around the Sr. Management table being that it would go on forever. It was for a time our most popular page on the site.



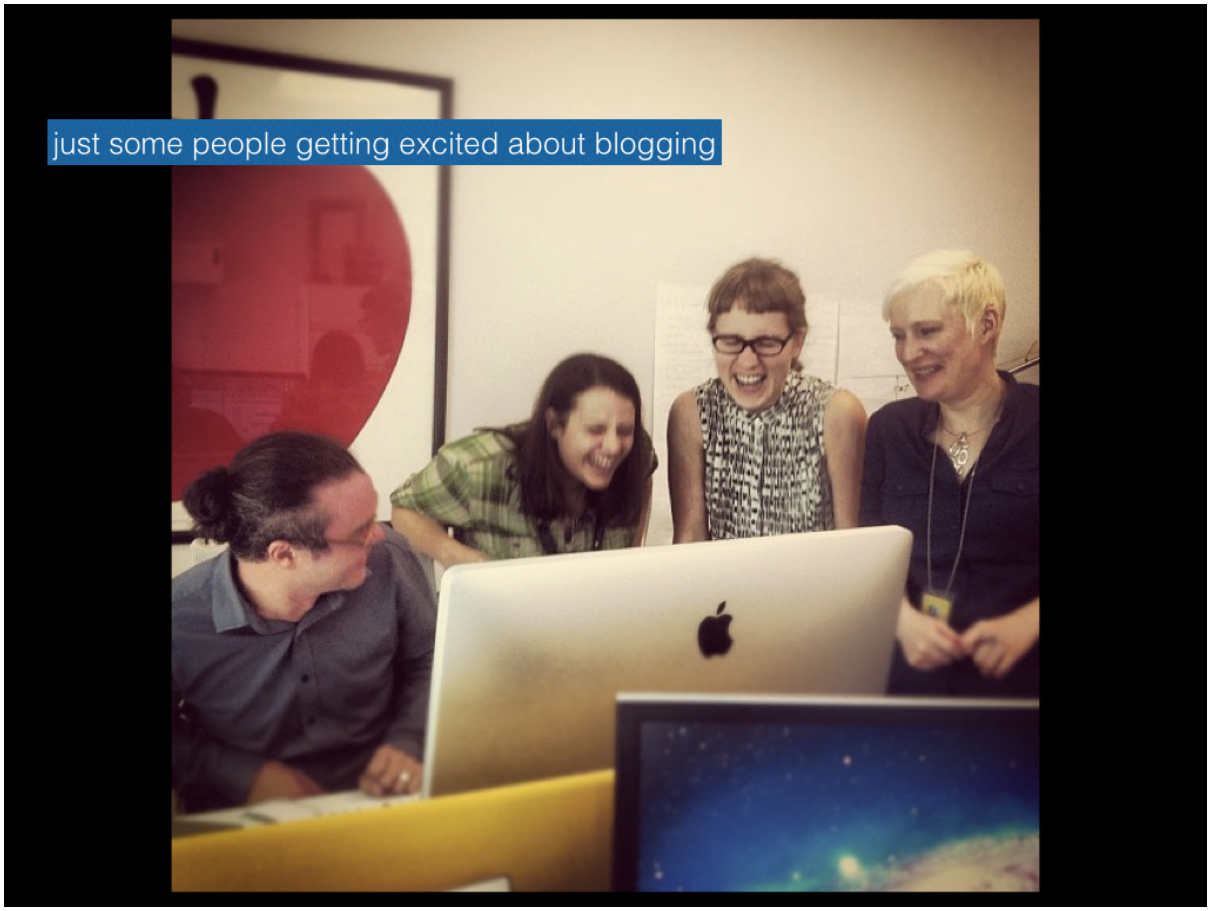
http://3ug4ii3uortgp68ee31cybxjyz.wpengine.netdna-cdn.com/wp-content/uploads/2014/04/MW2014.021.jpg

If there is one truth we have learned about maintaining a website using a CMS its that you'll eventually jump ship and switch to something else. In fact you may do this operation again and again. Its just the nature of the beast—the grass is always greener.

When we realized we needed to jump ship, we took to heart all the feedback we got from our content creators. We realized that what they really wanted were pleasant, easy to work with tools that allowed them to feel empowered. Tools that gave them a sense of authority, and made them feel good about the work they were doing. Like it was a way for them to communicate with the world all the important things they had going on.

In the end we chose WordPress. We looked at lots of options. We thought about even simpler options like a static site generator, or hmm, Squarespace? Could a museum run their entire website on Tumblr? All of these options afforded us with a great user experience, but seemed to trade of the ability to be flexible enough for our institutions needs. It really depends on the needs of each institution.

We searched far and wide. But we kept coming back to WordPress. It was familiar to everyone. Many of the staff already had their own WordPress blogs. WordPress gave us a nice balance between having the ability to create a sophisticated website and also being simple enough to use. In fact, while I was writing tools to migrate our content to WordPress, we realized that its more simplified system allowed us to re-organize our content, making the site easier to navigate. It's not that we couldn't do this in Drupal, but over time, Drupal just got out of control, because it let us.



http://3ug4ii3uortgp68ee31cybxjyz.wpengine.netdna-cdn.com/wp-content/uploads/2014/04/MW2014.022.jpg

We realized through the Object of the Day project that it was our CMS that stood in the way of success. The content was already good, the audience was already there. We just needed a way to get our own staff excited about doing it. It shouldn't be hard. It should be really easy and really fun to do. WordPress lets our staff get excited about the work they are doing. It gives them a simple to use, enjoyable writing experience, and for the editors, we found some really great plugins that let them manage all the content without feeling overwhelmed. Thats really why we chose WordPress.

We kind of think of it as a downgrade on the technical side of things, but its definitely an up-

grade when it comes to usability.

The end.

Postscript

There was some good discussion following the talk. A few things of note that were brought up included how our staff already had some experience with WordPress via our DesignOther90.org http://designother90.org website, our use of EditFlow http://editflow.org/ for notifications and calendaring/scheduling of content and Pressbooks http://pressbooks.com/ to aid with the production of our eBooks.

> *Editflow and Pressbooks good plugin tools in WordPress CMS for editorial management. @micahwalter* https://twitter.com/micahwalter *#MW2014* https://twitter.com/search?q=%23MW2014&src=hash
>
> *— JPoleon (@ohiofoodlovers) April 5, 2014* https://twitter.com/ohiofoodlovers/statuses/452506763439636480

We also talked a little about hindsight…

> *Q: How would you have done things differently? A: I wouldn't have used drupal – @micahwalter* https://twitter.com/micahwalter *. Hindsight is a wonderful thing. #mw2014* https://twitter.com/search?q=%23mw2014&src=hash
>
> *— Dafydd James (@DafJames) April 5, 2014* https://twitter.com/DafJames/statuses/452504176074182656

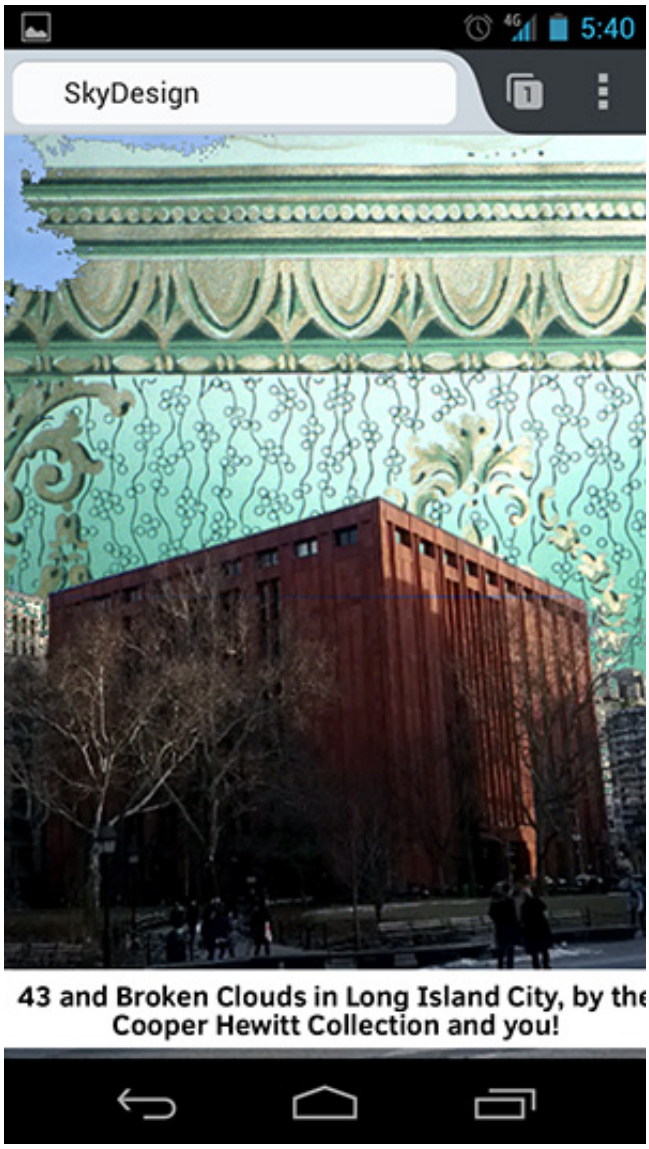
This entry was posted in Backends http://labs.cooperhewitt.org/category/backends/*, Conferences* http://labs.cooperhewitt.org/category/conferences/ *and tagged CMS* http://labs.cooperhewitt.org/tag/cms/*, drupal* http://labs.cooperhewit-

t.org/tag/drupal/, *MW2014* http://labs.cooperhewitt.org/tag/mw2014/, *wordpress* http://labs.cooperhewitt.org/tag/wordpress/ *on April 6, 2014* [http://labs.cooperhewitt.org/2014/downgrading-your-website-or-why-we-are-moving-to-wordpress/] *by micah* http://labs.cooperhewitt.org/author/micah/.

Announcing SkyDesigner! Sam Brenner joins the Labs

Greetings readers! My name is Sam and I'm the new Interactive Media Developer here at the Cooper-Hewitt's Digital and Emerging Media department. I'm thrilled to be here with the opportunity to help design and build the future of the museum, both online and in-house.

As part of my application for the position, I built SkyDesigner https://collection.cooperhewitt.org/play/skydesigner/, a web application that lets users replace the color of the sky with a picture of a similarly-colored object from the Cooper-Hewitt's collection. The "sky" idea comes from the original assignment, which was to create an application using both a weather API and the Cooper-Hewitt API https://collection.cooperhewitt.org/api/, but you can use SkyDesigner to swap out colors from anything you can take a picture of (meaning, it's great for selfies http://3ug4ii3uortgp68ee31cybxjyz.wpengine.netdna-cdn.com/wp-content/uploads/2014/05/Screenshot_2014-05-01-17-45-40.png). Give it a try now https://collection.cooperhewitt.org/play/skydesigner/!


5:40
SkyDesign
43 and Broken Clouds in Long Island City, by the
Cooper Hewitt Collection and you!



Here's how it works: first, users take a picture. If they're on a computer, they can use their webcam. If they're on a smartphone, they can use the built-in camera. Android users get (in my opinion) the better experience, because Android supports getUserMedia https://developer.-mozilla.org/en-US/docs/Web/API/Navigator.getUserMedia – this means that users can start their camera and take a picture without ever having to leave the application. iOS doesn't support getUserMedia http://caniuse.com/stream yet, so they are sent off to the native iOS camera app to take their picture, which then gets passed back to the browser. Once I receive the picture, I load it into a canvas.

In the next step, users tap on their picture to select a color. The color's hex code is sent straight to the Cooper-Hewitt API's search https://collection.cooperhewitt.org/api/methods/cooperhewitt.search.ob-

jects method, where I search for similarly-colored objects that have an associated image. While waiting for a response from the API, I also tell the canvas to make every pixel within range of the selected color become transparent http://www.hmp.is.it/creating-chroma-key-effect-html5-canvas/. When I get the image back from the API, I load it in behind the canvas and presto! It shows through where the selected color used to be. Finally, the image is titled based on the object's creator and your current weather information.

It's built using HTML, CSS and JavaScript. The original application had PHP to talk to the API but that's since been ported to JavaScript since I now have the luxury of running the site on the Collections website itself where we have our own built-in API hooks.

Being a weekend project, there are some missing features – sharing is a big one – but I think it demonstrates the API's ability to provide fresh, novel ways into a museum's vast collection. Here's the link again https://collection.cooperhewitt.org/play/skydesigner/, and you can also find the source on GitHub https://github.com/sambrenner/skydesigner.

This entry was posted in Experimental http://labs.cooperhewitt.org/category/experimental/ *and tagged computer vision* http://labs.cooperhewitt.org/tag/computer-vision/, *mashup* http://labs.cooperhewitt.org/tag/mashup/, *robot eyes* http://labs.cooperhewitt.org/tag/robot-eyes/ *on May 5, 2014* [http://labs.cooperhewitt.org/2014/announcing-skydesigner-sam-brenner-joins-the-labs/] *by Sam Brenner* http://labs.cooperhewitt.org/author/brenners/.

Making of: Design Dictionary Video Series

We often champion processes of iterative prototyping in our exhibitions and educational workshops about design. Practicing what we preach by actually adopting iterative prototyping workflows in-house is something we've been working on internally at Cooper Hewitt for the last few years.

In the 3.5 years that I've been here, I've observed some inspiring progress on this front. Here's one story of iterative prototyping and inter-departmental collaboration in-house, this time for our new Design Dictionary web video series.

Design Dictionary is a 14-part video series https://www.youtube.com/playlist?list=PLqwPGOOIhKSDEqOg8OxfciHIMTfyJa9vu that aims to demystify everything from tapestry weaving to 3D printing in a quick and highly visual way. With this project, we aimed not only to produce a fun and educationally valuable new video series, but also to shake up our internal workflow.

Content production isn't the first thing you'd think of when discussing iterative prototyping workflows, but it's just as useful for media production as it is for hardware, software, graphic design, and other more familiar design processes.

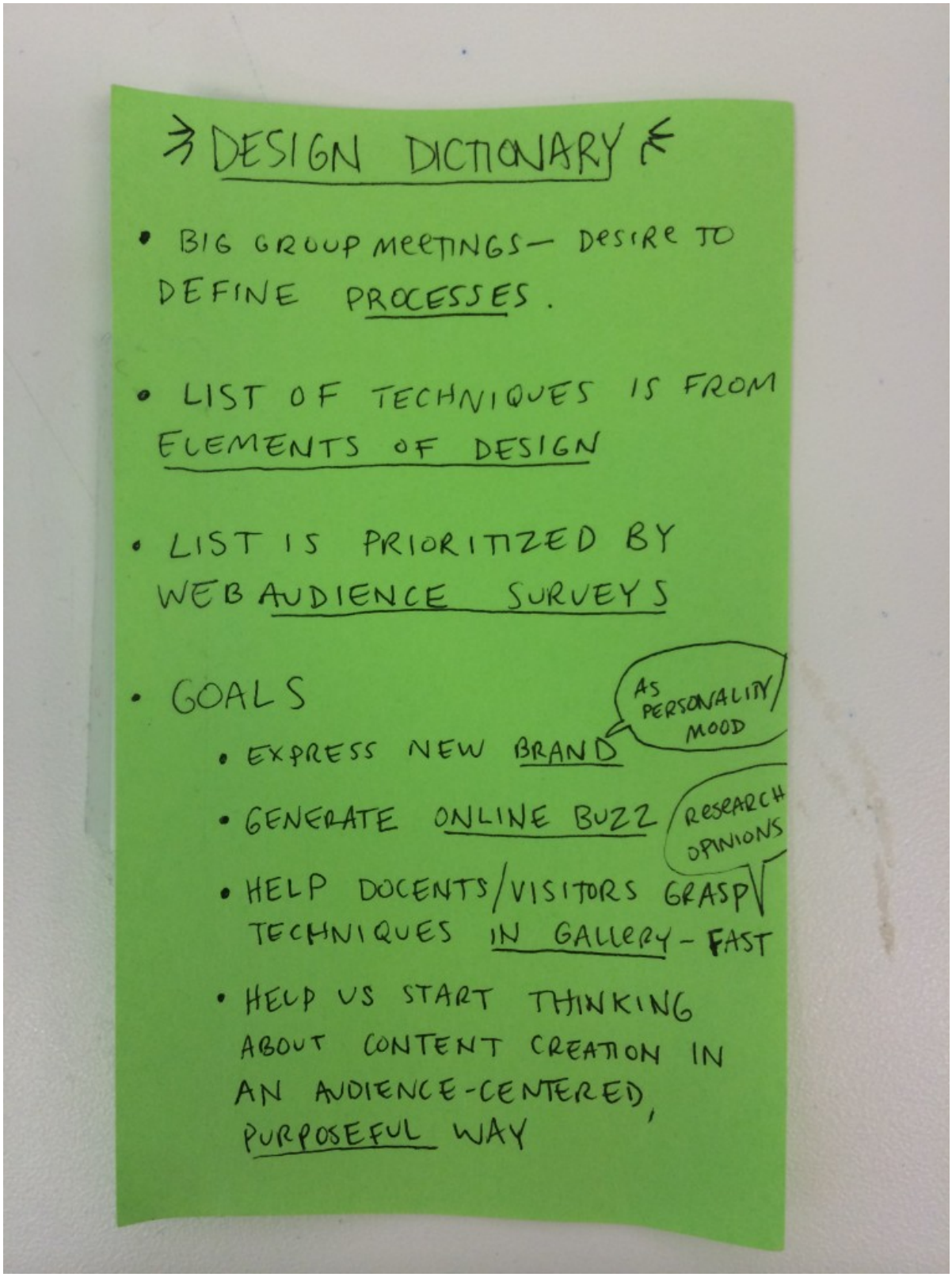
The origin of Design Dictionary traces back to a new monthly meeting series that was kicked off about two years ago. The purpose of the meetings was to get Education, Curatorial, and Digital staff in the same room to talk about the content being developed for our new permanent collection exhibition, *Making Design* http://www.cooperhewitt.org/events/opening-exhibitions/. We wanted everything from the wall labels to the digital interactive experiences to really resonate with our various audiences. Though logistically clunkier and more challenging than allowing content development to happen in a small circle, big-ish monthly meetings held the promise of diverse points of view and the potential for unexpected and interesting ideas.

At one of these meetings, when talking about videos to accompany the exhibition, the curators and educators both expressed a desire to illustrate the various design techniques employed in our collection via video. It was noted that video of most any technique is already available online, but since these videos are of varying quality, accuracy, and copyright allowances, and it might be worth it to produce our own series.

I got the ball rolling by creating a list of techniques that will appear more than once in *Making Design.*

Then I collected a handful of similar videos online, to help center the conversation about project goals. Even the habitual "lurkers" http://en.wikipedia.org/wiki/Lurker on Basecamp http://en.wikipedia.org/wiki/Basecamp_Classic were willing to chime in when it came to criticizing other orgs' educational videos: "so boring!" "so dry!" they said. This was interesting, because as a media producer it can be hard to 1) get people to actually participate and submit their thoughts and 2) break it to someone that their idea for a new video is extremely boring.

Once we were critiquing *somebody else's* educational videos, and not our own darling ideas, people seemed more able to see video content from a viewer's perspective (impatient, wanting excitement) as opposed to a curator/educator's perspective (fixated on detail, accuracy, thoroughness, less concerned with the viewer's interests & attention span).



http://3ug4ii3uortgp68ee31cybxjyz.wpengine.netdna-cdn.com/wp-content/uploads/2014/06/photo-e1403532912321.jpg

I kept this note taped to my screen as a reminder of the 4 project goals.

It is amazingly easy to get confused and lost mid-project if you don't keep your goals close. This is why I clung tightly to the sticky note shown above. When everyone involved can agree on goals up-front, the project itself can shape-shift quite nicely and organically, but the goals stay firm. Stakeholders' concerns can be evaluated against the goals, not against your org. hierarchy or any other such evil criteria.

Even with all the viewer-centric empathy in the world, it can still be hard to predict what your audience will like and dislike. Would a video about tapestry weaving get any views on YouTube? What about 3D printing?



http://3ug4ii3uortgp68ee31cybxjyz.wpengine.netdna-cdn.com/wp-content/uploads/2014/06/Screen-Shot-2014-06-23-at-10.39.10-AM.png

We asked our Twitter followers which techniques interest them most.

We created a quick survey on SurveyMonkey and blasted it out https://twitter.com/cooperhewitt/status/398098901854457857 to our followers on Facebook and Twitter to gauge the temperature.

1. Which techniques would you like to know more about? (Choose 4)

Create Chart | Download

	Response Percent	Response Count
Lithography	31.1%	14
Weaving	22.2%	10
Jacquard Weaving	11.1%	5
Knitting	8.9%	4
3D Printing	**55.6%**	**25**
Glassblowing	31.1%	14
Felting	13.3%	6
Screen Printing	26.7%	12
Laser Cutting	51.1%	23
Ceramics	22.2%	10
Lacemaking	28.9%	13
Engraving	22.2%	10
Etching	15.6%	7
Embroidery	17.8%	8
Paper-making	17.8%	8
Tapestry	24.4%	11
Something else: Hide Responses		5

Responses (5) | Text Analysis | My Categories (0)

Filter by Category ▼ | Categorize As ▼ | Search Responses

Showing 5 text responses | No responses selected

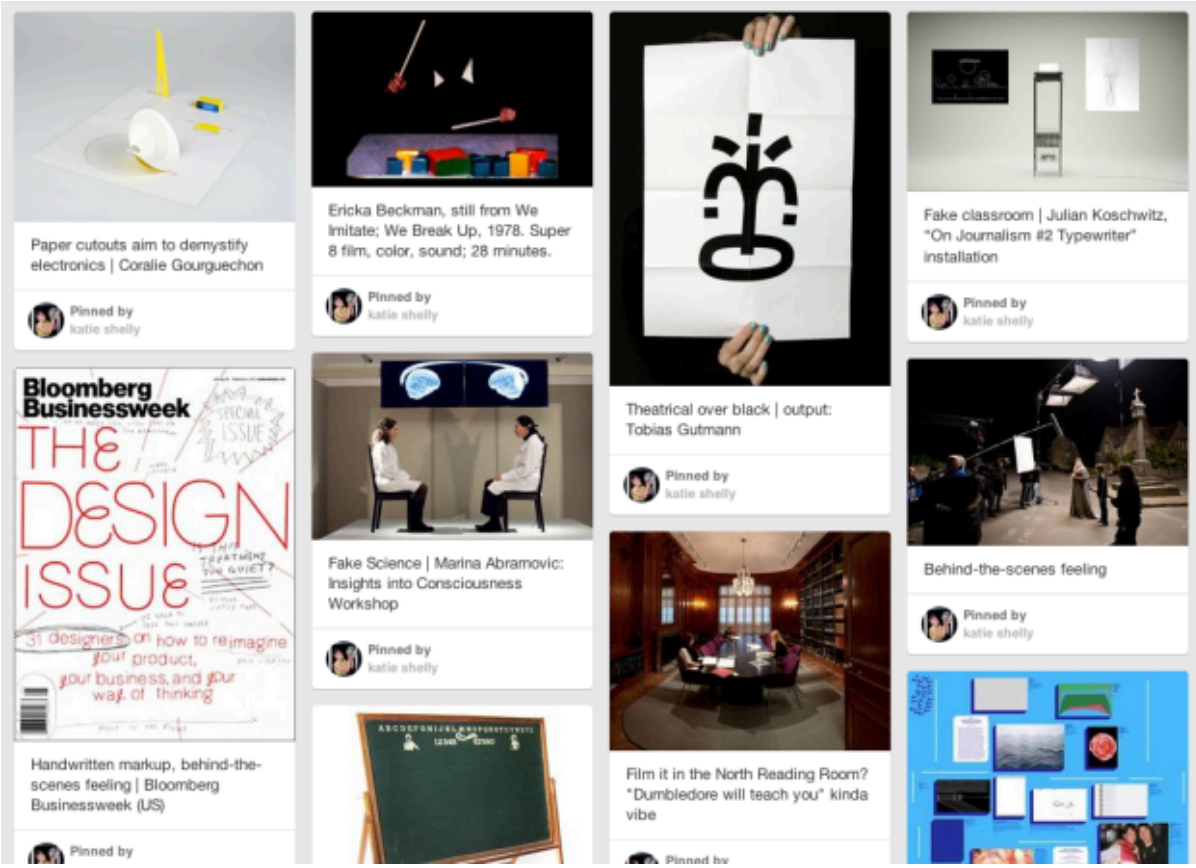
- Medieval manuscript design
 11/12/2013 10:06 AM View Responses Categorize As ▼
- Historic fashion
 11/12/2013 10:02 AM View Responses Categorize As ▼
- handwritten calligraphy
 11/6/2013 2:00 PM View Responses Categorize As ▼
- Woodworking
 11/6/2013 1:23 PM View Responses Categorize As ▼
- silver smithing
 11/6/2013 12:03 AM View Responses Categorize As ▼

http://3ug4ii3uortgp68ee31cybxjyz.wpengine.netdna-cdn.com/wp-content/uploads/2014/06/Techiques-Survey.png

Surveying our Twitter and Facebook fans with SurveyMonkey, to learn which techniques they'd be interested in learning more about.

We also hosted the same survey on Qualaroo https://qualaroo.com/, which pops up on our website. My hunch about what people would say was all wrong. We used these survey results to help choose which techniques would get a video.

By this point, it was mid-winter 2014, and our new brand from Pentagram http://www.underconsideration.com/brandnew/archives/new_logo_and_identity_for_cooper_hewitt_by_pentagram.php#.U6g--41dVYM was starting to get locked in. It was a good opportunity to play with the idea of expressing this new brand via video. What should the pacing and rhythm be like? How should animations feel? What kind of music should we use?



http://3ug4ii3uortgp68ee31cybxjyz.wpengine.netdna-cdn.com/wp-content/uploads/2014/06/Screen-Shot-2014-06-23-at-10.00.30-AM.png

Public mood-boarding with Pinterest.

Seb & I are fans of "Look Around You" https://www.youtube.com/watch?v=t4CRCJUmWsM and we liked the idea of

a somewhat cheeky approach to the dreaded “educational video.” How about an educational video that (lovingly, artfully) mocks the very format of educational videos? I created a Pinterest board to help with the art direction http://www.pinterest.com/interkatie/design-process-shorts/. We couldn’t go too kitsch with the videos, however, because our new brand is pretty slick and that would have clashed.

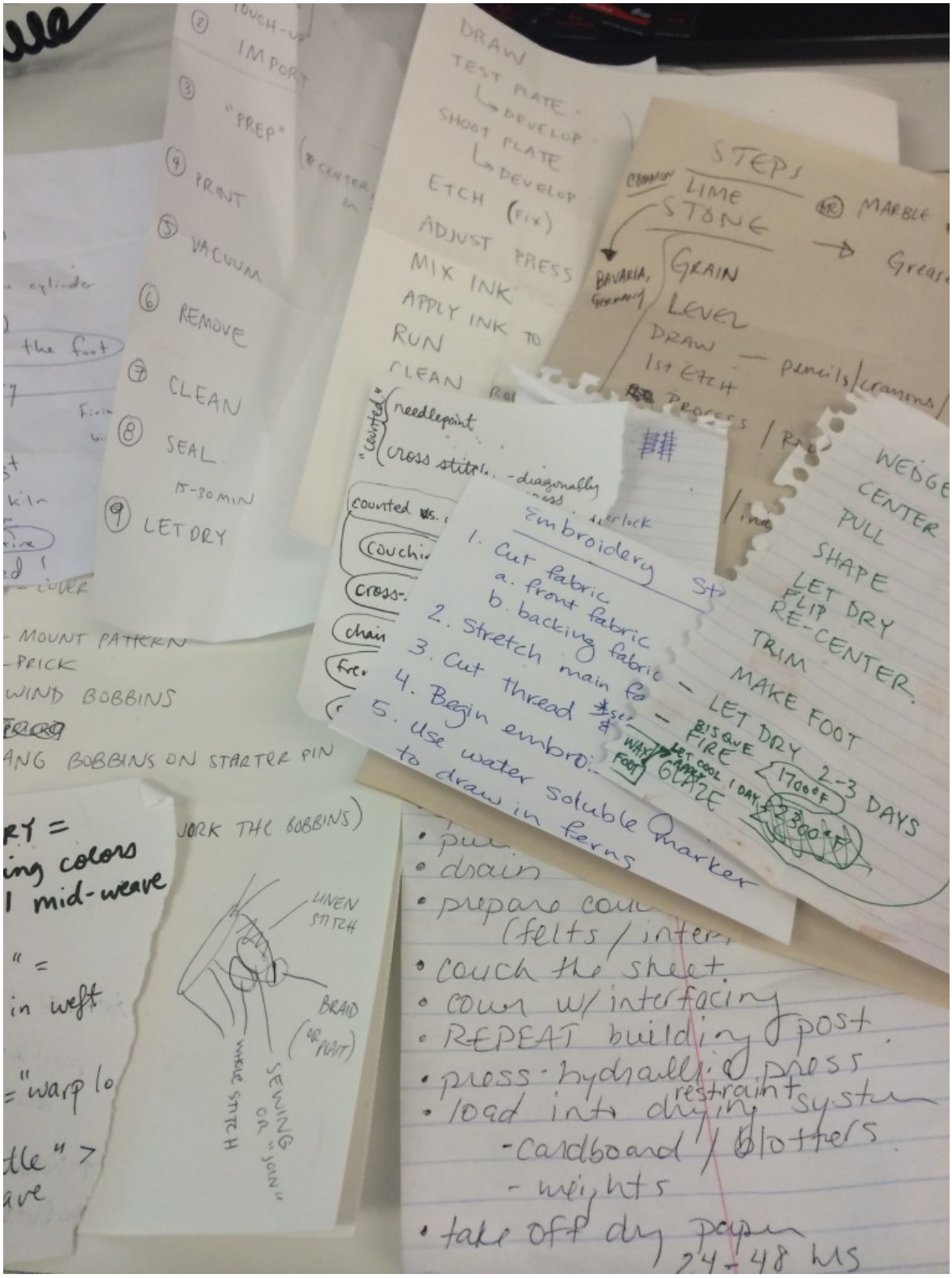
Then I made a low-stakes, low-cost prototype, recycling footage from a previous project. I sent this out to the curatorial/education team for feedback using Basecamp.

In retrospect I can now see that this video is awful. But at the time, it seemed pretty good to me. This is why we prototype, people!

With feedback from colleagues via Basecamp (less book, more live action, more prominent type), I made the next prototype:

I got mixed reactions about the new typography. Some found it distracting. And I was still getting a lot of mixed reactions to the book. So here was my third pass:

I was starting to reach out to artists and designers to lend their time to the shoots, and was cycling that fresh footage into the project, and cycling the new video drafts back to the group for feedback. Partially because we were on a deadline and partially because it works well in iterative projects, we didn’t wait for closure on step 1 before moving on to step 2.



http://3ug4ii3uortgp68ee31cybxjyz.wpengine.netdna-cdn.com/wp-content/uploads/2014/06/photo-1-e1403532881605.jpg

I got a crash course in 14 different techniques.

Every new shoot presented a new chance to test the look and feel and get reactions from my colleagues. Here was a video where I tried my own hand at graphical “annotations” (dovetail, interlock, slit):

By this point my prototype was refined enough to share with Pentagram, who were actively working on our digital collateral. I asked them to style a typographic solution for the series, which could serve as the basis for other museum videos as well. Whenever you can provide a designer with real content, do it, because it’s so much better than using dummy content. Dummy content is soft and easy, allowing itself to be styled in a way that looks good, but meets no real requirements when put through a real stress test (long words, bulky text, realistic quantities of donor credits, real stakeholders wanting their interests represented prominently).

Here is a revised video that takes Pentagram’s new, crisp typography into account:

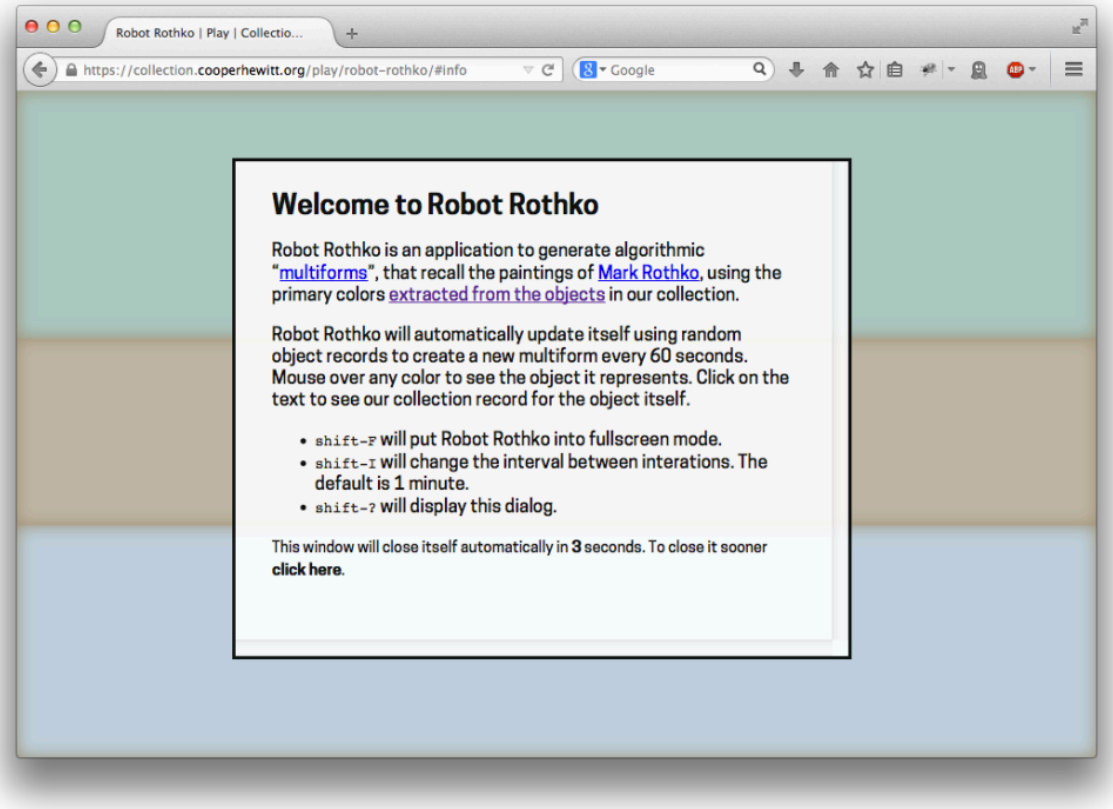
This got very good feedback from education and curatorial. And I liked it too. Yay.

All-in-all, it took about 8 rounds of revision to get from the first cruddy prototype to the final polished result.

And here are the final versions.

This entry was posted in Publishing http://labs.cooperhewitt.org/category/publishing/ *and tagged Content* http://labs.cooperhewitt.org/tag/content/, *prototyping* http://labs.cooperhewitt.org/tag/prototyping/, *video* http://labs.cooperhewitt.org/tag/video/ *on June 23, 2014* [http://labs.cooperhewitt.org/2014/making-of-design-dictionary-vid-series/] *by katieshelly* http://labs.cooperhewitt.org/author/katieshelly/.

Robot Rothko



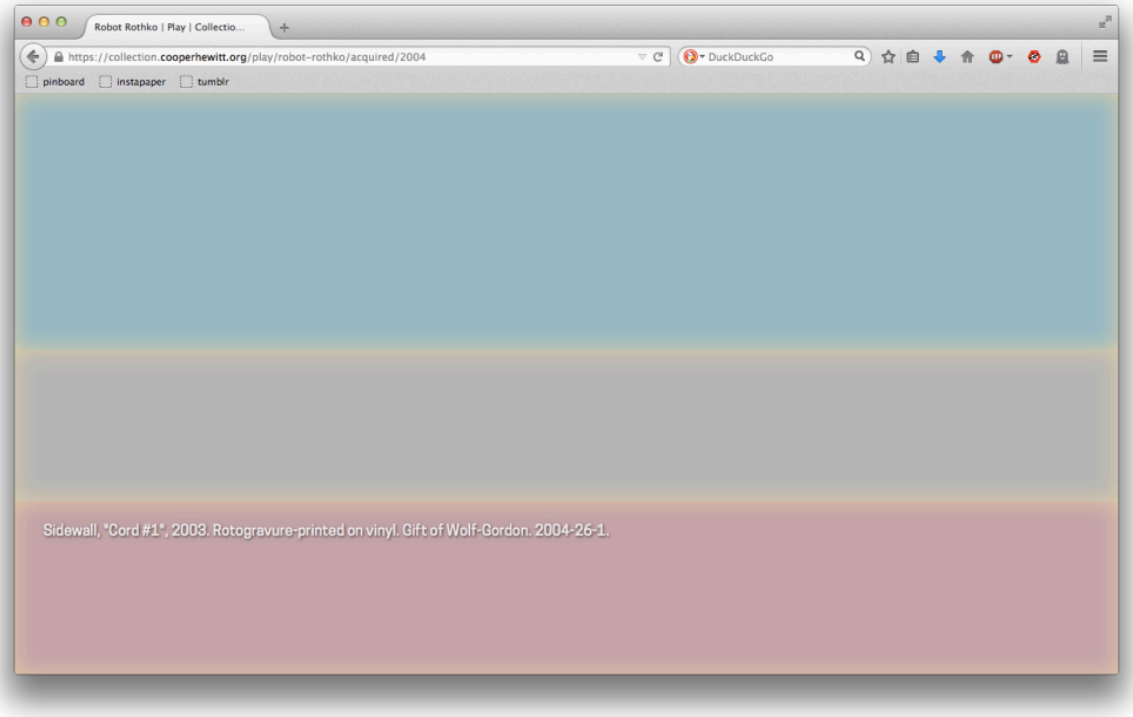
https://collection.cooperhewitt.org/play/robot-rothko/#info

> *Now that I've written this blog post it occurs to me that it would be trivial to build something similar on top of the Cooper Hewitt Collections API* https://collection.cooperhewitt.org/api/ *— since that's ultimately where all this colour stuff* http://labs.cooperhewitt.org/2013/giv-do/ *comes from — so I will probably do that shortly and stick in it the Play* https://collection.cooperhewitt.org/play/ *section.*

That's something I wrote last week http://www.aaronland.info/weblog/2014/05/12/non/#multiforms on my personal weblog. I was writing about a little web "application" that I'd made to generate algorithmic "multiforms http://en.wikipedia.org/wiki/Mark_Rothko#Rothko.27s_.22multiforms.22 " that recall the work of the late painter Mark Rothko http://www.markrothko.org/ . The source of the colors used to create these robot-

multiforms are derived from photo uploads and extracted using the same code that the Cooper Hewitt uses to generate color palettes for the objects in our collection. We wrote about that process last year. http://labs.cooperhewitt.org/2013/giv-do/

These robot "paintings" are built by fetching three photos and using their primary color to fill one of three stacked rectangles that make up the canvas. A dominant color for a fourth photo is used along with an inset CSS3 box-shadow https://developer.mozilla.org/en-US/docs/Web/CSS/box-shadow to give the illusion a fuzzy, hazy background on which the rectangles sit. Every 60 seconds a new version is generated and the colors (and boxes) gently transition from old to new.

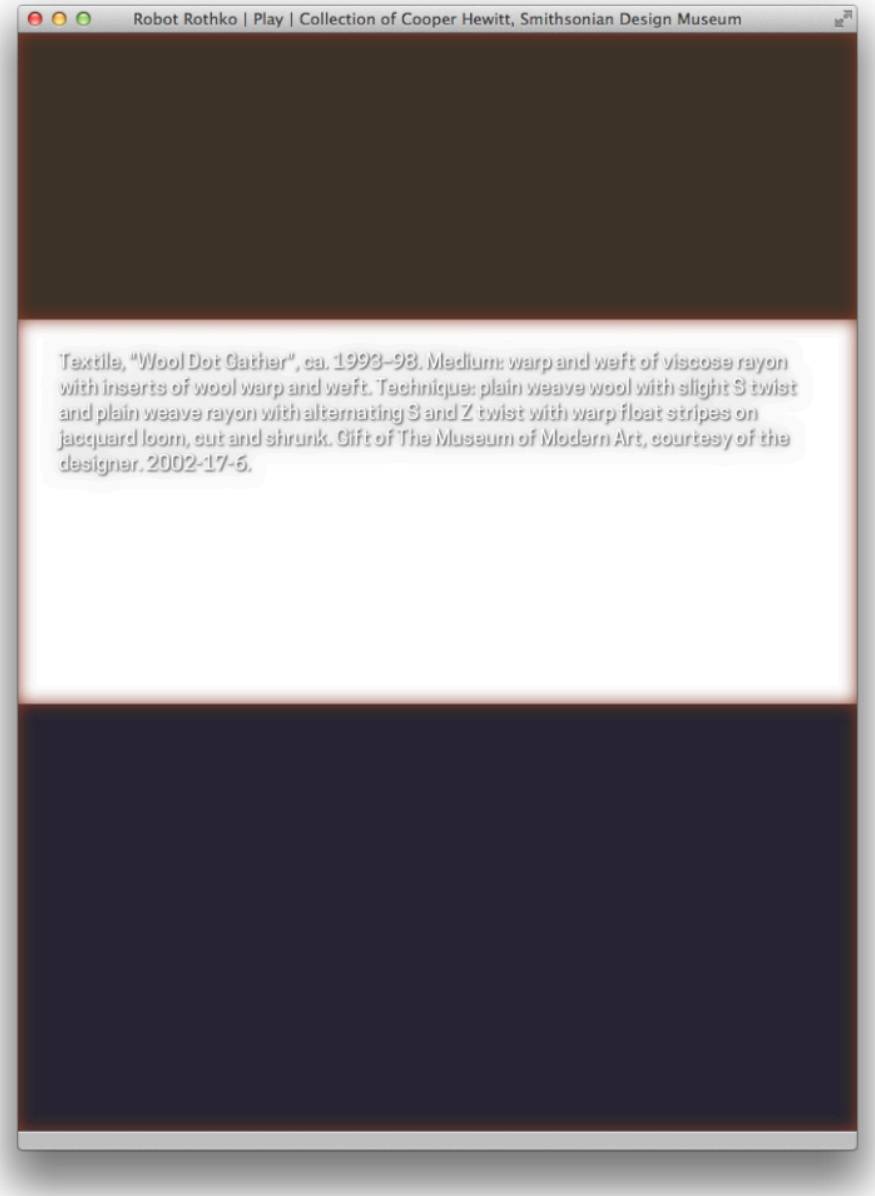


https://collection.cooperhewitt.org/play/robot-rothko/acquired/2004/

In that original blog post, I also wrote:

> *That's it. It doesn't do anything else and that's part of the charm for me. It just sits in the background running in second-screen-mode* http://russelldavies.typepad.com/planning/2011/06/secondary-attention-part-one-optics.html *stamping out robot-Rothko paintings. … It's nice to have a new screen friend* http://booktwo.org/notebook/thamestide/ *to spend the days the days with*

They're not really Rothko paintings, obviously, and to suggest that they are would do the painter a disservice. Rothko's paintings are not just any random set of colors stacked on top of one another. Rothko worked long and hard to choose the arrangement of his paintings and it's easy to imagine that he would have been horrified by some of the combinations that Robot Rothko offers up. But like the experimental Albers Boxes feature http://labs.cooperhewitt.org/2013/albers-boxes/ they are a nod and gesture – and a wink – towards the real thing.



https://collection.cooperhewitt.org/play/robot-rothko/

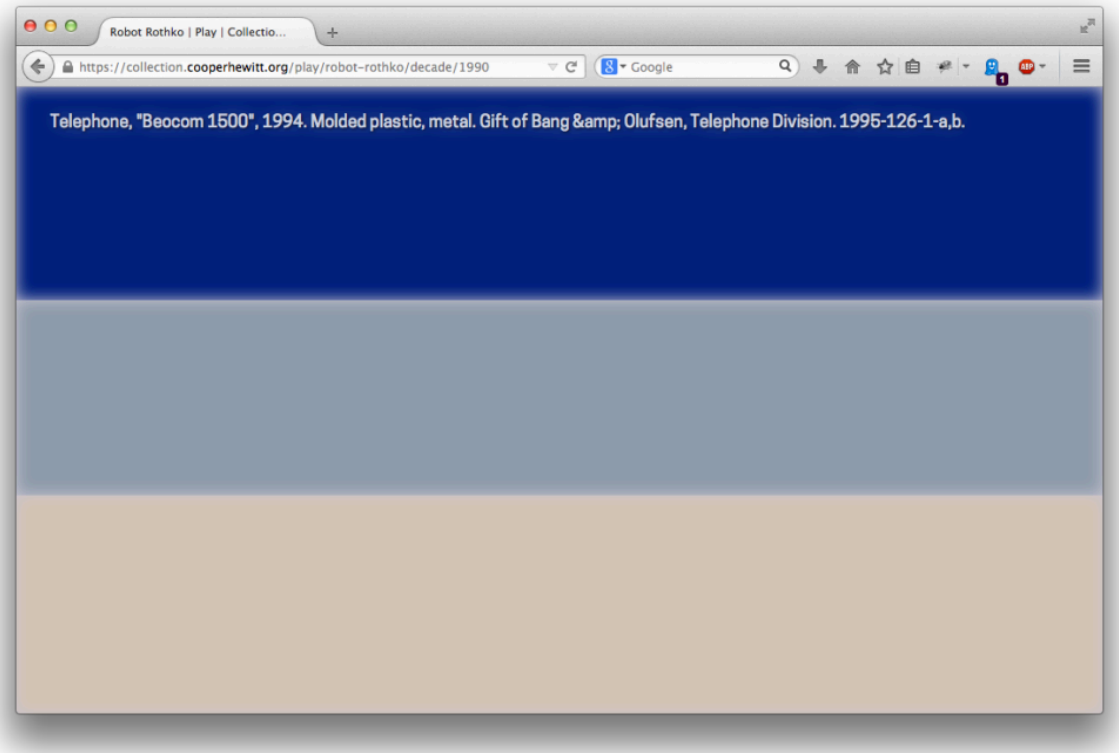
Having gotten things working for a personal non-museum and not-really-for-strangers project I decided that it would be nice to do something similar for for the museum *which is absolutely for everyone*. So, today we are launching Robot Rothko https://collection.cooperhewitt.org/play/robot-rothko/#info which is exactly the same as the application described above except that it uses objects from our collection http://collection.cooperhewitt.org/objects/ instead of photos as its source material. Like this:

https://collection.cooperhewitt.org/play/robot-rothko/#info https://collection.cooperhewitt.org/play/robot-rothko/#info

See the **#info** part of that URL? That will cause the application to load with an information box that explaining what you're looking at (and that will close itself automatically after 30 seconds). If you just want to jump straight to the application all you have to do is remove the **#info** from the URL.

https://collection.cooperhewitt.org/play/robot-rothko/ https://collection.cooperhewitt.org/play/robot-rothko/#info

Robot Rothko will automatically update itself using random object records to create a new multiform every 60 seconds. Mouse over any color to see the object it represents. Click on the text to see our collection record for the object itself.



https://collection.cooperhewitt.org/play/robot-rothko/decade/1990/

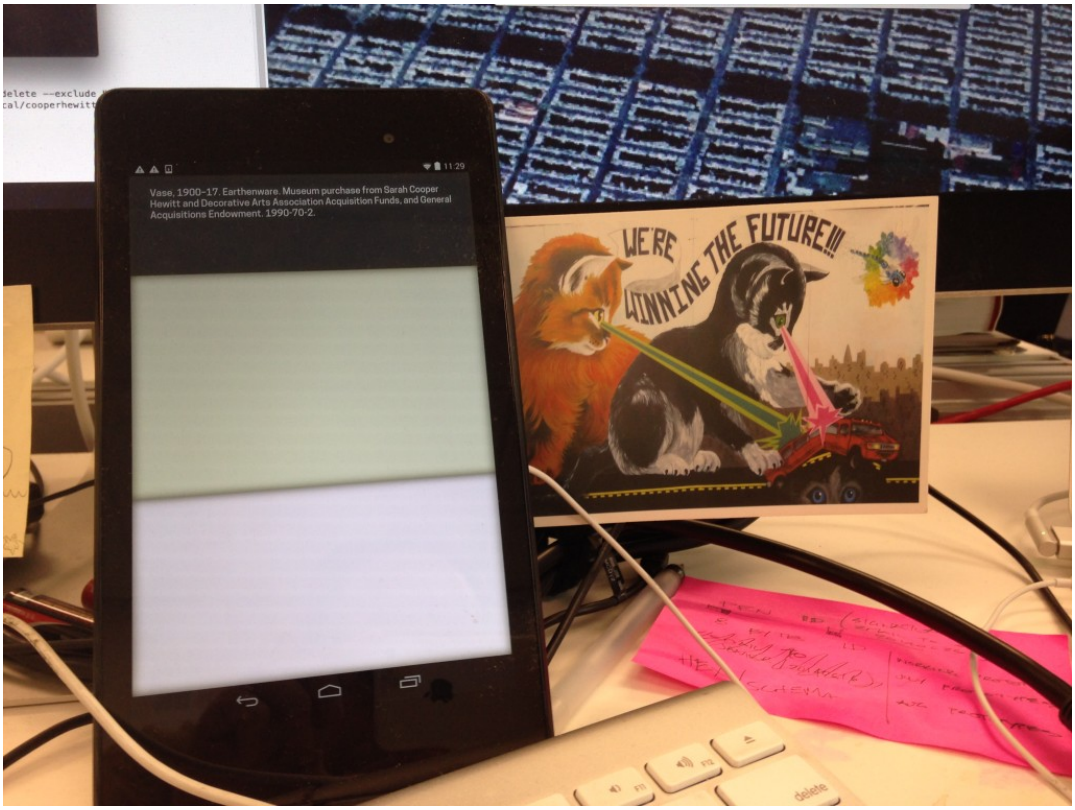
You can also filter stuff by person, decade. You can also filter by the year we acquired an object if you can guess where it is; that one still feels a little buggy so we're going to hold off publishing the URL until we can figure out what's wrong. Here are some examples of the first two:

https://collection.cooperhewitt.org/play/robot-rothko/people/18046041 https://collection.cooperhewitt.org/play/robot-rothko/people/18046041

https://collection.cooperhewitt.org/play/robot-rothko/decade/1910 https://collection.cooperhewitt.org/play/robot-rothko/decade/1910

Robot Rothko is native to the web https://archive.org/details/NativeToAWebOfData which means it will work in any modern web browser whether it's on your desktop or your phone or your tablet. It can be put it to fullscreen mode (by pressing shift-F) and if you save the website's URL to your homescreen on your phone, or tablet, it is configured to launch without any of the usual browser chrome. If you use a Mac you can plug the URL for Robot Rothko in to Todd Ditchendorf's handy Fluid.app http://fluidapp.com/ which will turn it all in to a shiny desktop application. /

am guessing there are equivalent tools for Windows or Linux but I don't know what they are.



If you'd like to generate your own Robot Rothkos http://collection.cooperhewitt.org/play/robot-rothko/ there's an API method for doing just that:

https://collection.cooperhewitt.org/api/methods/cooperhewitt.play.robotRothko https://collection.cooperhewitt.org/api/methods/cooperhewitt.play.robotRothko

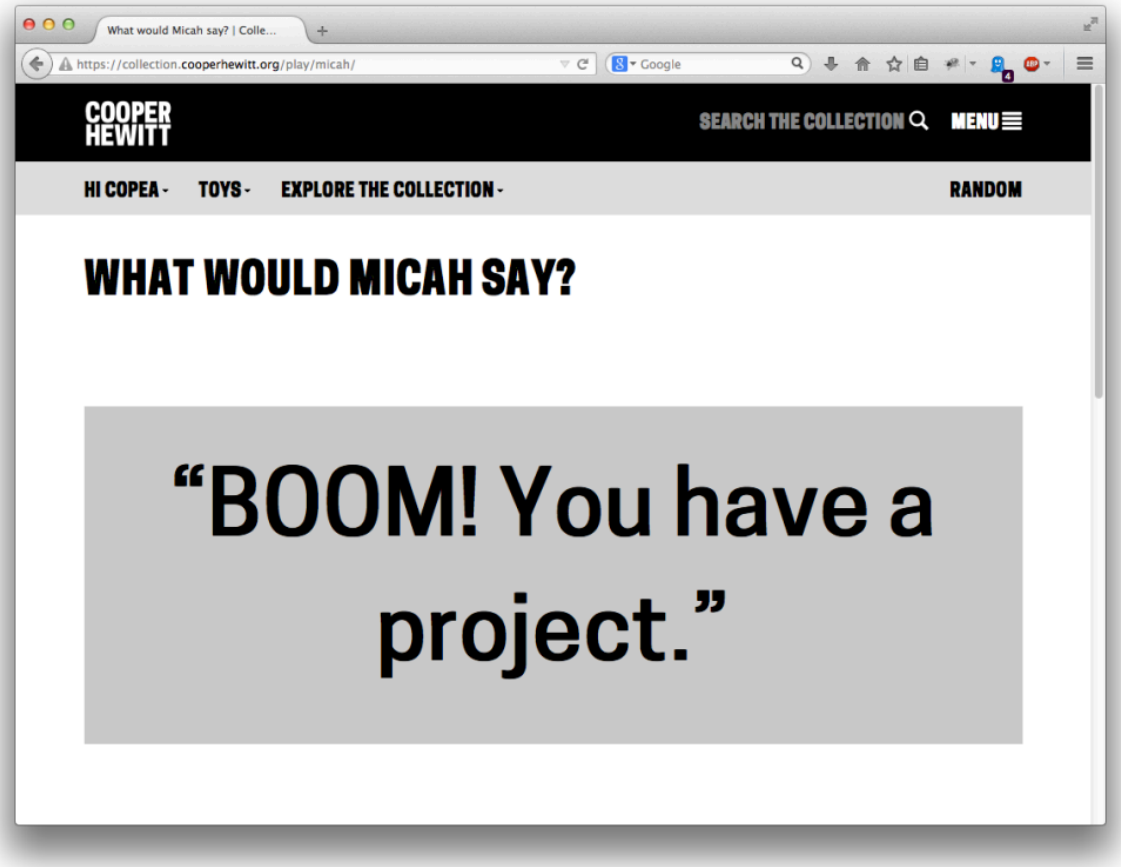
And of course it works with our recently announced support for DSON https://collection.cooperhewitt.org/api/formats/dson/ as a response format:

```
curl -X GET 'https://api.collection.cooperhewitt.org/rest/?method=cooper-
hewitt.play.robotRothko&access_token=SEEKRET&person_id=18041501&for-
mat=dson'
```

```
such "rothko" is such "canvas" is so "49" and "28" and "23" many and
"palette" is so such "colour" is "#b8ab5b" , "id" is "18805769" , "epi-
```

```
taph" is "Folding Fan, 1900\u201305. Medium: silk, wood, horn, metal,
metal spangles. Gift of Lillian C. Hart. 1985-89-1." wow ? such "colour"
is "#c7c7c7" . "id" is "18640557" ! "epitaph" is "Drawing, "Two Studies
for Rectangul", ca. 1965. Pen and black ink on white wove paper. Gift of
Vladimir Kagan. 1992-56-7." wow , such "colour" is "#db8952" , "id" is
"18133219" , "epitaph" is "Fragment, mid-18th century. Medium:
silk\nTechnique: plain weave patterned by supplementary warp floats and
complementary weft floats. Gift of John Pierpont Morgan. 1902-1-811." wow
many ? "background" is such "colour" is "#c7a9af" . "id" is "18761047" !
"epitaph" is "Booklet Cover Sheet, 1916. Color woodcut on lavender wove
paper paper. Museum purchase from Drawings and Prints Council Fund and
through gift of Margery and Edgar Masinter and Merrill C. Berman. 1999-
50-1-3." wow wow , "filters" is so many and "stat" is "ok" wow
```

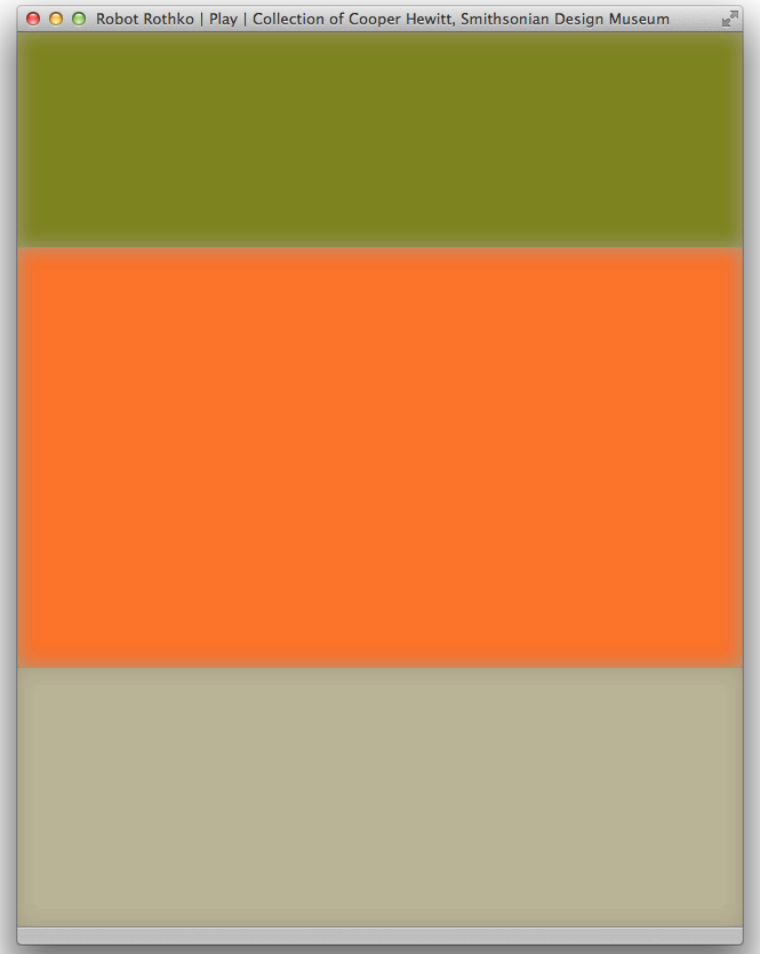
Robot Rothko lives in a new section of the collections website called "Play http://collection.cooperhewitt.org/play". The distinction between the Play section and the Experimental Features http://collection.cooperhewitt.org/experimental/ section of the website can probably be easiest thought of as: Experimental features are things that apply to the entirety of the collections website, while Play things are small contained applications that use the collections API http://collection.cooperhewitt.org/api and focus on or build off a particular aspect of the collection. The first of these was Sam Brenner's SkyDesigner http://labs.cooperhewitt.org/2014/announcing-skydesigner-sam-brenner-joins-the-labs/ and Robot Rothko is actually the third such application.



http://collection.cooperhewitt.org/play/micah/

In between those two was What Would Micah Say? https://collection.cooperhewitt.org/play/micah/ (WWMS) a quick end-of-day project to test out the W3C's Text-to-Speech APIs http://www.w3.org/community/speech-api/ that are starting to appear in some web browsers (read: Chrome and Safari as of this writing, and make sure you have the volume turned up). The WWMS "application" was mostly a simple 20-minute exercise to test whether fetching some content dynamically and feeding to the text-to-speech APIs actually works and produces something useable. It does, which is very exciting because it opens up any number of accessibility-related improvements we can starting thinking about adding to the collections website.

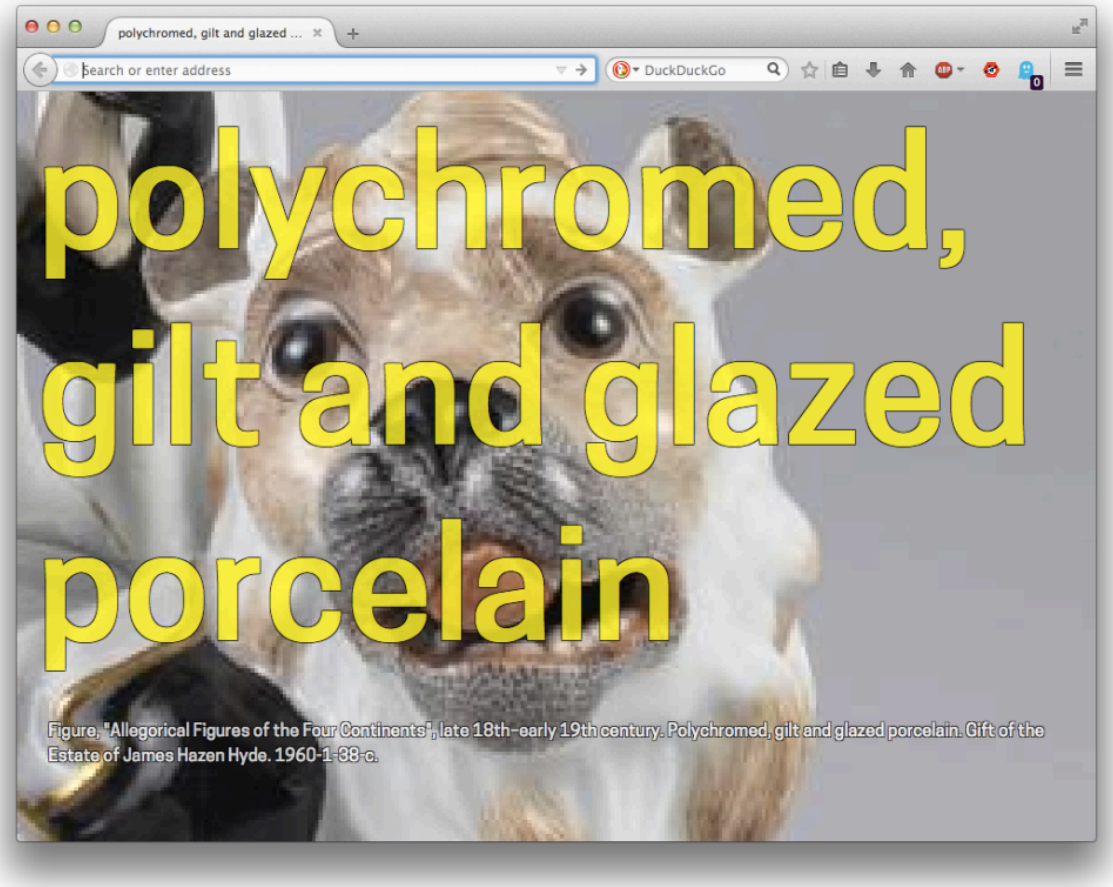
That we happened to use the cooperhewitt.labs.whatWouldMicahSay API method https://collection.cooperhewitt.org/api/methods/cooperhewitt.labs.whatWouldMicahSay and then configured the text-to-speech API to read his words as if spoken by a "French" robot made it all a little bit silly and a little more fun but those are important considerations. Because sometimes playing at – or making interesting – a technical problem is the best way to work through whether it is even worth pursuing in the first place.



https://collection.cooperhewitt.org/play/robot-rothko/people/18046041

This entry was posted in Experimental http://labs.cooperhewitt.org/category/experimental/ *and tagged API* http://labs.cooperhewitt.org/tag/api/, *colors* http://labs.cooperhewitt.org/tag/colors/, *robot eyes* http://labs.cooperhewitt.org/tag/robot-eyes/ *on July 7, 2014* [http://labs.cooperhewitt.org/2014/robot-rothko/] *by asc* http://labs.cooperhewitt.org/author/asc/.

The Medium is the Message (and pubsock-etd)



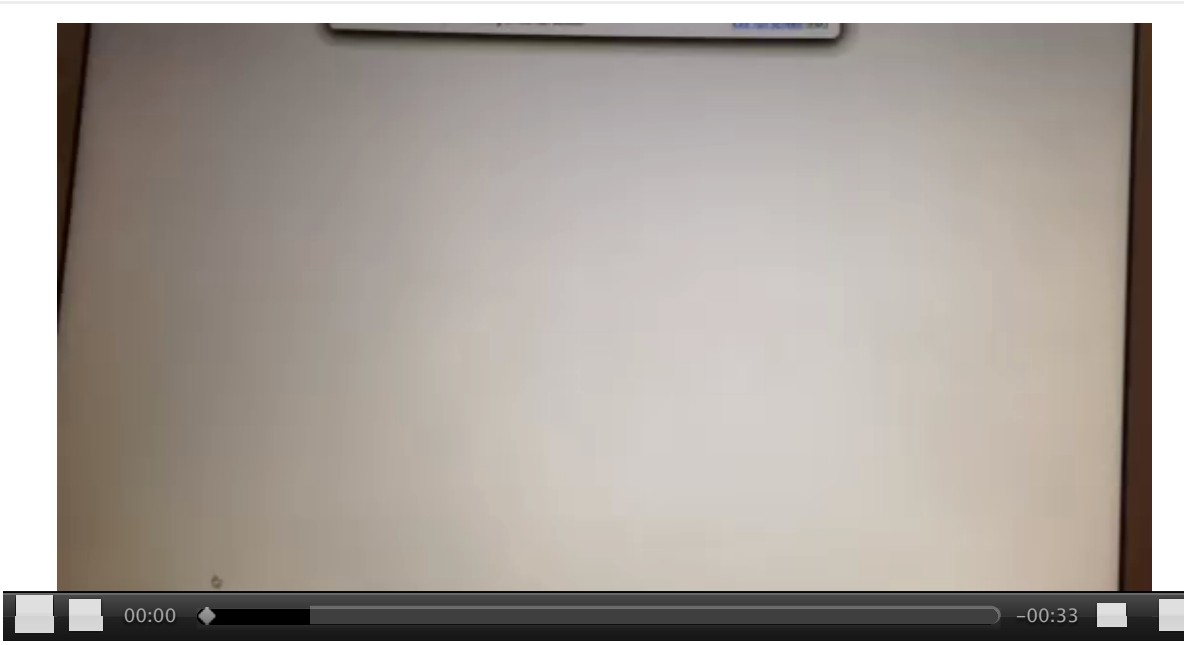
http://3ug4ii3uortgp68ee31cybxjyz.wpengine.netdna-cdn.com/wp-content/uploads/2014/08/Screen-Shot-2014-08-02-at-1.31.57-PM.png

Have you ever wanted to see a real-time view of all the objects that people are looking at on the collections website http://collection.cooperhewitt.org/ ? Now you can!

At least for objects with images. There are lots of opportunities to think about interesting ways to display objects without images but since everything that follows has been a weekends-and-mornings project we've opted to start with the "simple" thing first.

We have lots of different ways of describing media: 12, 865 ways https://collection.cooperhewitt.org/media at last count to be precise. The medium with the most objects (2, 963) associated with it is

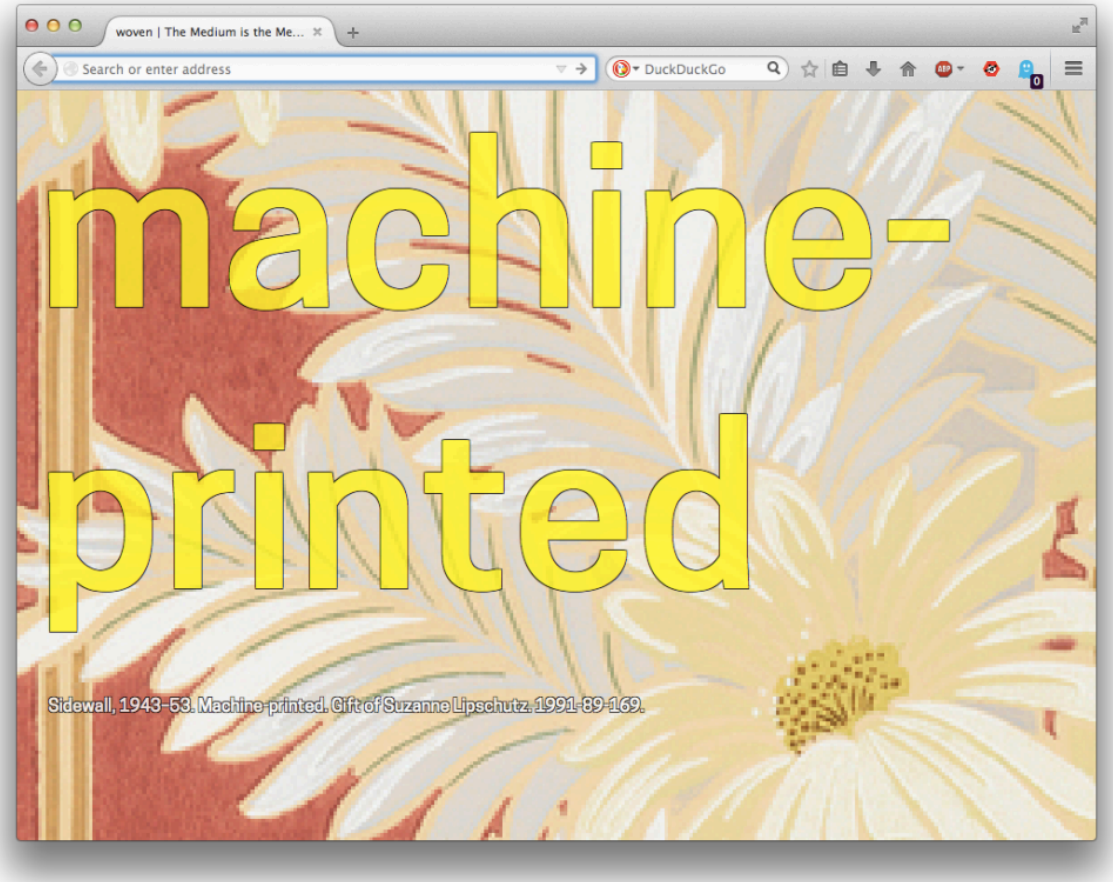
cotton https://collection.cooperhewitt.org/media/35402927/ but all of these numbers are essentially misleading. The history of the cataloging of the collection has preferenced precision and detail over the kind of rough bucketing (for example, tags) that lots of people are used to these days.



It's a practice that can sometimes seem frustrating in the moment but, in the long-run, we're better served for it. In time we will get around to assigning high-level categorizations for equally high-level browsing but it's worth remembering that the practice of describing objects in minute detail predates things like databases, which we take for granted today. In fact these classifications, and their associated conventions and rituals, were the de-facto databases before computers or databases had even been invented.

But 13,000 different media, most of which only describe a single object, can be overwhelming. Where do you start? How do you know what to look for? Given the breadth of our collection what *don't* we have? And given the level of detail we try to assign to objects how to do you whether a search doesn't yield any results because it's not in our collection or simply because we're using a different name for the same thing you're looking for?

This is a genuinely Big and Hairy Problem and we have not solved it yet. But the ability to relay objects as they are viewed by the public, in real-time, offers an interesting opportunity: *What if we just displayed (and where possible, read aloud) the medium for that object?*



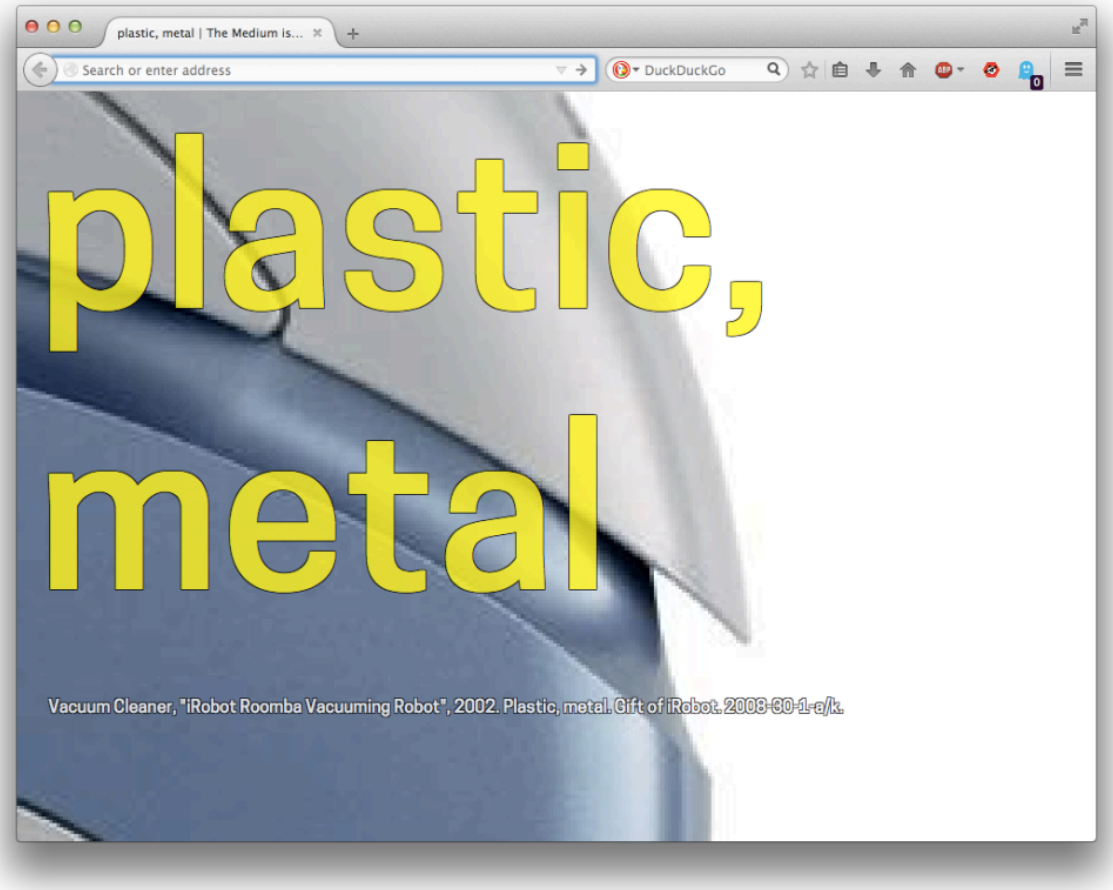
http://3ug4ii3uortgp68ee31cybxjyz.wpengine.netdna-cdn.com/wp-content/uploads/2014/08/Screen-Shot-2014-08-02-at-1.31.29-PM.png

That's all The Medium is the Message http://medium.collection.cooperhewitt.org/#info does: It is an ambient display that let you keep an eye on the kinds of things that are in our collection and offered a gentle, polite way to start to see the shape of all the different things that tell the story of the museum. It's not a tool to help you take a quiz so much as a way to absorb an awareness of the collection as if by osmosis. To show people an *aspect* of the collection as an avenue to begin understanding its entirety.

> *We're not thinking enough about sound. If we want all these things to communicate with us, and we don't want to be starting at screens and they're going to do more than flash a couple of lights, then we need to work with sound. Either 'sound effects' that mean something or devices that talk to us. Personally, I think it'll be the latter morphing into the former. And this is worth thinking about because it's already creeping up on us. Self-serve checkouts are talking at us, reversing trucks are beeping at us, trucks turning left are barking at us, incoherently – all with much less apparent thought and 'design' than we devote to screens.*

— *Russell Davies,* *the internet of talking* http://russelldavies.typepad.com/planning/2014/07/internetoftalking.html

While The Medium is the Message http://medium.collection.cooperhewitt.org/#info is a full-screen application that displays a scaled-up version of the square-crop thumbnail for an object http://labs.cooperhewitt.org/2013/b-is-for-beta/ it also tries to use your browser's text-to-speech capabilities to read aloud that object's medium. It may not be the kind of thing you want playing in a room full of people but alone in your room, or under a pair of headphones, it's fun to imagine it as a kind of Music for Airports https://en.wikipedia.org/wiki/Music_for_Airports for cultural heritage.



http://3ug4ii3uortgp68ee31cybxjyz.wpengine.netdna-cdn.com/wp-content/uploads/2014/08/Screen-Shot-2014-08-02-at-1.32.27-PM.png

Text-to-speech http://www.w3.org/community/speech-api/ is currently best supported in Chrome and Safari. Conversely the best support for crisp and pixelated image-rendering https://developer.mozilla.org/en-US/docs/Web/CSS/image-rendering is in Firefox. Because… computers, right?

For the time being The Medium is the Message http://medium.collection.cooperhewitt.org/ lives in a little sand-box all by itself over here:

http://medium.collection.cooperhewitt.org http://medium.collection.-cooperhewitt.org/#info

Eventually we hope to merge it back in to the main collections website http://collection.cooperhewitt.org/play/ but since it's all brand-new we're going to put it some place where it can, if necessary, have little melt-downs and temper-tantrums without adversely affecting the rest of the collections website. It's also worth noting that some internal networks – like at a big company or organization – might still disallow WebSockets traffic which is what we're using for this. If that's the case try waiting until you're home.



http://3ug4ii3uortgp68ee31cybxjyz.wpengine.netdna-cdn.com/wp-content/uploads/2014/08/Screen-Shot-2014-08-04-at-3.16.31-PM.png

And now, for the Nerdy Bits: The rest of this blog post is captial-T technical so you can stop reading now if that's not your thing (though we think it's stil pretty interesting even if the details sound like gib-

berish).

The Medium is the Message http://medium.collection.cooperhewitt.org/ is part of a larger project to investigate a few different tools in order to understand how they might fit together and to what effect. They are:

- Redis http://redis.io/ and in particular its implementation of Publish/Subscribe messaging paradigm http://redis.io/topics/pubsub – Every once in a while there's a piece of software that is released which feels like genuine magic. Arguably one of the last examples of this was memcached https://en.wikipedia.org/wiki/Memcached originally written by Brad Fitzpatrick, for the website LiveJournal and without which entire slices of the web as we now know it wouldn't exist. Both Redis and memcached are similar in spirit in that their feature-set is limited by design but what they claim to do Just Works™ and both have broad support across the landscape of programming languages. That last piece is incredibly important since it means we can use Redis to bridge applications written in whatever language suits the problem best. We'll return to that idea in the discussion of "step 0" below.
- Websockets https://developer.mozilla.org/en-US/docs/WebSockets – WebSockets are a way for a web browser and a server to create and maintain a persistent connection and to shuttle messages back and forth. Normally the chatter between a browser and a server happens akin to the way two people might send each other postcards in the mail and WebSockets are more like a pair of teenagers calling each on the phone and talking for hours and hours and hours. Sort of like Pub/Sub for a web browser, right? WebSockets have been around for a few years now but they are still a bit of a new territory; super-cool but not without some pitfalls.
- Go http://golang.org/ – Go is a programming language from the nice people at Google, that recently celebrated its fourth anniversary. It is part of growing trend in language design to find a middle ground between loosely typed languages, and the need to develop stable applications with a minimum of fussiness. Go is probably not the language we would develop a complex user-facing application in but for long-running services with well-defined boundaries it seems kind of perfect. (Go's notion of code-based channels http://www.golang-book.com/10/index.htm are a fascinating parallel to both Pub/Sub and WebSockets but that's a whole other blog post.)

Fun fact: The Labs' very own Sam Brenner http://www.samjbrenner.com/ 's ITP thesis project called Adventures of Teen Bloggers http://samjbrenner.com/notes/the-adventures-of-teen-bloggers/ is an archive of old LiveJournal accounts in the shape of an 8-bit video game!



http://teenbloggers.net/

In order to test all of those technologies and how they might play together we built pubsocketd https://github.com/cooperhewitt/go-pubsocketd/ which is a simple daemon written in Go that subscribes to a Pub/Sub channel and ferries those messages to a browser using Websockets (WS).

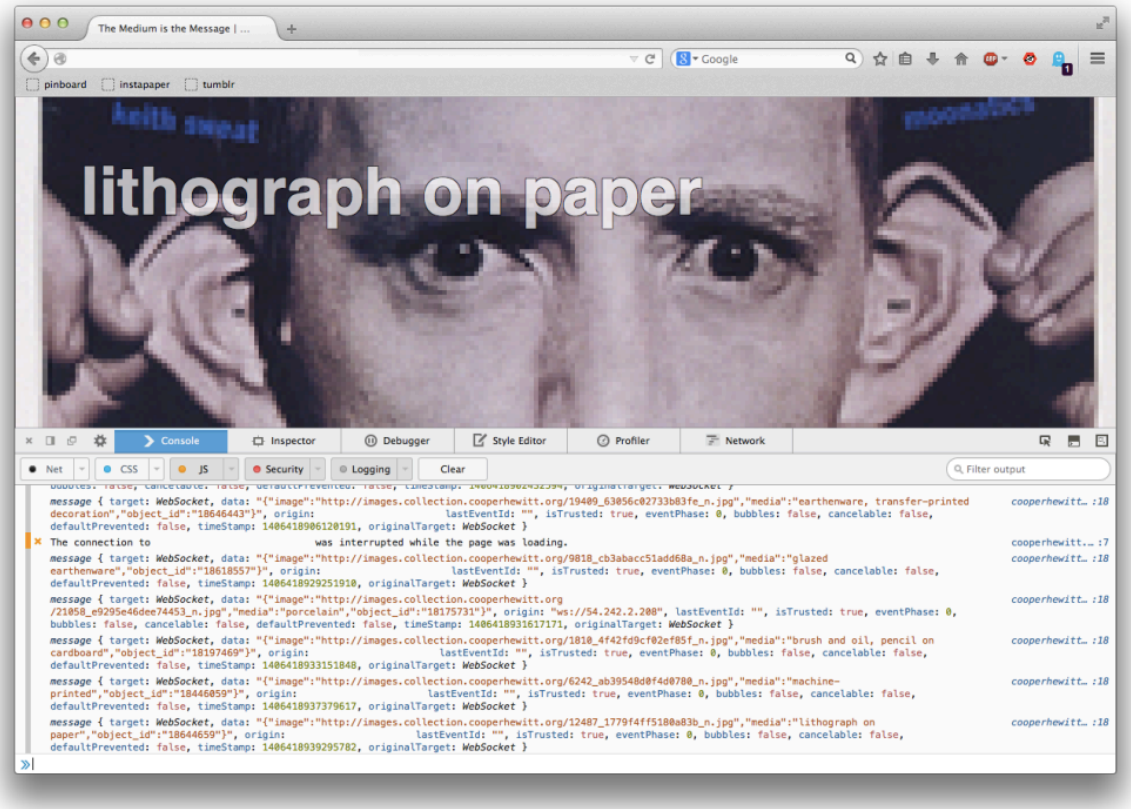
1. Listen for messages from a specific (Redis) Pub/Sub channel
2. Accept incoming WS requests
3. Shuttle any messages from the Pub/Sub channel to all the open WS connections

That's it. It is left up to WS clients (your web browser) to figure out what to do with those messages.

```
$> ./pubsocketd -ws-origin=http://example.com
2014/08/01 17:23:38 [init] listening for websocket re-
quests on 127.0.0.1:8080/, from http://example.com
2014/08/01 17:23:38 [init] listening for pubsub mes-
sages from 127.0.0.1:6379 sent to the pubsocketd channel
2014/08/01 17:23:44 [10.20.30.40][10.20.30.40:56401][handshake] OK
```

```
2014/08/01 17:23:44 [10.20.30.40][10.20.30.40:56401][request] OK
2014/08/01 17:23:44 [10.20.30.40][10.20.30.40:56401][connect] OK
2014/08/01 17:23:53 [10.20.30.40][10.20.30.40:56401][send] OK
2014/08/01 17:24:05 [10.20.30.40][10.20.30.40:56401][send] OK
# and so on...
```

The “step 0” in all of this is the ability for the collections website itself to connect to a Redis server and *send* a Pub/Sub message, whenever someone views an object, to the same channel that the pubsocketd server is listening to.



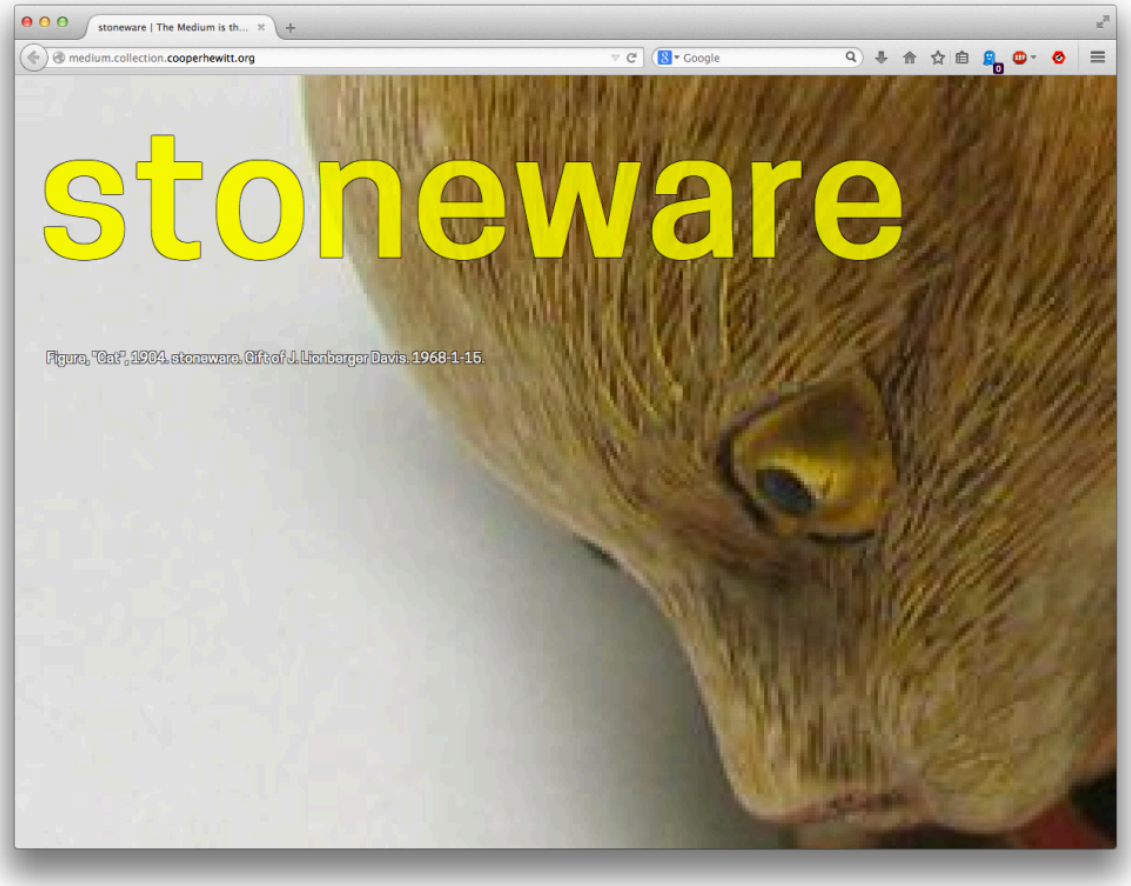
http://www.dischord.com/fugazi_live_series

This allows for a nice clean separation of concerns and provides a simple way for related, but fundamentally discrete, applications to interact without getting up in each other’s business.

Given the scope of the project we probably could have accomplished the same thing, with less scaffolding, using Server-Sent Events (SSE) https://developer.mozilla.org/en-US/docs/Server-sent_events/Using_server-sent_events but this was as much an exercise designed to get our feet wet with both WebSockets

and Go so it's been worth doing it the "hard way".

Matthew Rothenberg http://www.mroth.info/, creator of the popular EmojiTracker http://www.emojitracker.com/, was nice enough to open-source the Go-based SSE server endpoint https://github.com/mroth/sseserver he wrote to feed his application and we may eventually re-write The Medium is the Message http://medium.collection.cooperhewitt.org/, or future applications like it, to use that.



http://3ug4ii3uortgp68ee31cybxjyz.wpengine.netdna-cdn.com/wp-content/uploads/2014/08/Screen-Shot-2014-08-04-at-4.11.45-PM.png

We've open-sourced the code for pubsocketd https://github.com/cooperhewitt/pubsocketd/ under a BSD license and we welcome suggestions, patches and (gentle) clue-bats:

https://github.com/cooperhewitt/go-pubsocketd https://github.com/cooperhewitt/go-pubsocketd/

Enjoy!

This entry was posted in Backends http://labs.cooperhewitt.org/category/backends/, *Experimental* http://labs.cooperhewitt.org/category/experimental/ *on August 4, 2014* [http://labs.cooperhewitt.org/2014/the-medium-is-the-message-and-pubsocketd/] *by asc* http://labs.cooperhewitt.org/author/asc/.

Rethinking Search on the Collections Site

One of my longer-term projects since joining the museum has been rethinking how the search feature functions on the collections website. As we get closer to re-opening the museum with a suite of new technologies http://www.cooperhewitt.org/2014/06/16/reopening-press-releases/, our work in collaboration with Local Projects http://localprojects.net/ has prompted us to take a close look at the moving pieces that comprise the backend of our collections site and API. Search, naturally, forms a large piece of that. Last week, after a few weeks of research and experimentation, I pushed the first iteration live http://collection.cooperhewitt.org/search/collection. In this post, I'll share some of the thoughts and challenges that informed our changes.

First, a glossary of terms for readers who (like me, a month ago) have little-to-no experience with the inner-workings of a search engine:

- **Platform:** The software that actually does the searching. The general process is that we feed data to the platform (see "index"), and then we ask it for results matching a certain set of parameters (see "query"). Everything else is handled by the platform itself. Part of what I'll get into below involves our migration from one platform, Apache Solr http://lucene.apache.org/solr/, to another, Elasticsearch http://elasticsearch.org/.
- **Index:** An index is the database that the search platform uses to perform searches on. The search index is a lot like the primary database (it probably could fill that role if it had to) but it adds extra functionality to facilitate quick and accurate retrieval of search results.
- **Query:** The rules to follow in selecting things that are appropriate to provide as search results. For users, the query could be something like "red concert poster," but we have to translate that into something that the search provider will understand before results can be retrieved. Search providers give us a lot of different ways we can query things (ranges of a number, geographic distance or word matching to name a few), and a challenge for us as interface designers is to decide how transparent we want to make that translation. Queries also allow us to define how results should be sorted and how to facet results.
- **Faceting/Aggregation:** A way of grouping results based on traits they posses. For example, faceting on "location" when you search our collection for "cat" reveals that 80 of our cat-related things are from the USA, 16 are from France, and so on.
- **Analysis (Tokenization/Stemming etc):** A process that helps a computer work with sentences. Tokenization, for example, would split a search for "white porcelain vase" into

> the individual tokens: “white,” “porcelain” and “vase,” and then perform a search for any number of those tokens. Another example is stemming, which would allow the platform to understand that if a user searches for “running,” then items containing other words like “run” or “runner” are also valid search results. Analysis also gives us the opportunity to define custom rules that might include “marathon” and “track” as valid results in a search for “running.”

The State of Search

Our old search http://collection.cooperhewitt.org/search/ functionality showed its symptoms of underperformance in a few ways. For example, basic searches — phrases like “red concert poster” — turned up no results despite the presence of such objects in our collection, and searching for people would not return the person themselves, only their objects. These symptoms led me to identify what I considered the two big flaws in our search implementation.

On the backend, we were only indexing objects. This meant that if you searched for “Ray Eames,” you would see all of the objects we have associated with her, but to get to her individual person page, you would have to first click on an object and then click on her name. Considering that we have a lot of non-objects[1] #f1, it makes sense to index them all and include them, where relevant, in the results. This made my first objective to find a way to facilitate the indexing and querying of different types of things.

On the frontend, we previously gave users two different ways to search our collection. The default method, accessible through the header of every page, performed a full text search on our Solr index and returned results sorted by image complexity. Users could also choose the “fancy search” option, which allows for searches on one or more of the individual fields we index, like “medium,” “title,” or “decade.” We all agreed here that “fancy search” was confusing, and all of its extra functionality — faceting, searching across many fields — shouldn’t be seen as “advanced” features. My second objective in rethinking how search works, then, was to unify “fancy” and “regular” search into just “search.”

Objective 1: Update the Backend

Our search provider, Solr, requires that a schema be present for every type of thing being indexed. The schema (an XML file) tells Solr what kind of value to expect for a certain field and what sort of analysis to perform on the field. This means I’d have to write a schema file — anticipating how I’d like to form all the indexed data — for each new type of thing we want to

search on.

One of the features of Elasticsearch is that it is "schemaless," meaning I can throw whatever kind of data I want at the index and it figures out how to treat it. This doesn't mean Elasticsearch is always correct in its guesses — for example, it started treating our accession numbers as dates, which made them impossible to search on — so it also gives you the ability to define mappings, which has the same effect as Solr's schema. But if I want to add "people" to the index, or add a new "location" field to an object, using Elasticsearch means I don't have to fiddle with any schemas. This trait of Elasticsearch alone made worth the switch (see Larry Wall's first great virtue of programmers http://threevirtues.com/, laziness: "the quality that makes you go to great effort to reduce overall energy expenditure") because it's important to us that we have the ability to make quick changes to any part of our website.

Before building anything in to our web framework, I spent a few days getting familiar with Elasticsearch on my own computer. I wrote a python script that loops through all of the CSVs from our public collections repository and indexed them in a local Elasticsearch server. From there, I started writing queries just to see what was possible. I was quickly able to come up with a lot of the functionality we already have on our site (full-text search, date range search) and get started with some complex queries as well ("most common medium in objects between 1990-2000," for example, which is "paper"). This code is up on Github, so you can get started with your own Cooper Hewitt search engine at home https://github.com/cooperhewitt/collection-elasticsearch!

Once I felt that I had a handle on how to index and query Elasticsearch, I got started building it into our site. I created a modified version of our Solr indexing script (in PHP) that copied objects, people, roles and media from MySQL and added them to Elasticsearch. Then I got started on the endpoint, which would take search parameters from a user and generate the appropriate query. The code for this would change a great deal as I worked on the frontend and occasionally refactored and abstracted pieces of functionality, but all the pieces of the pipeline were complete and I could begin rethinking the frontend.

Objective 2: Update the Frontend

Updating the frontend involved a few changes. Since we were now indexing multiple categories of things, there was still a case for keeping a per-category search view that gave users access to each field we have indexed. To accommodate these views, I added a tab bar across the top of the search forms, which defaults to the full-collection search. This also eliminates

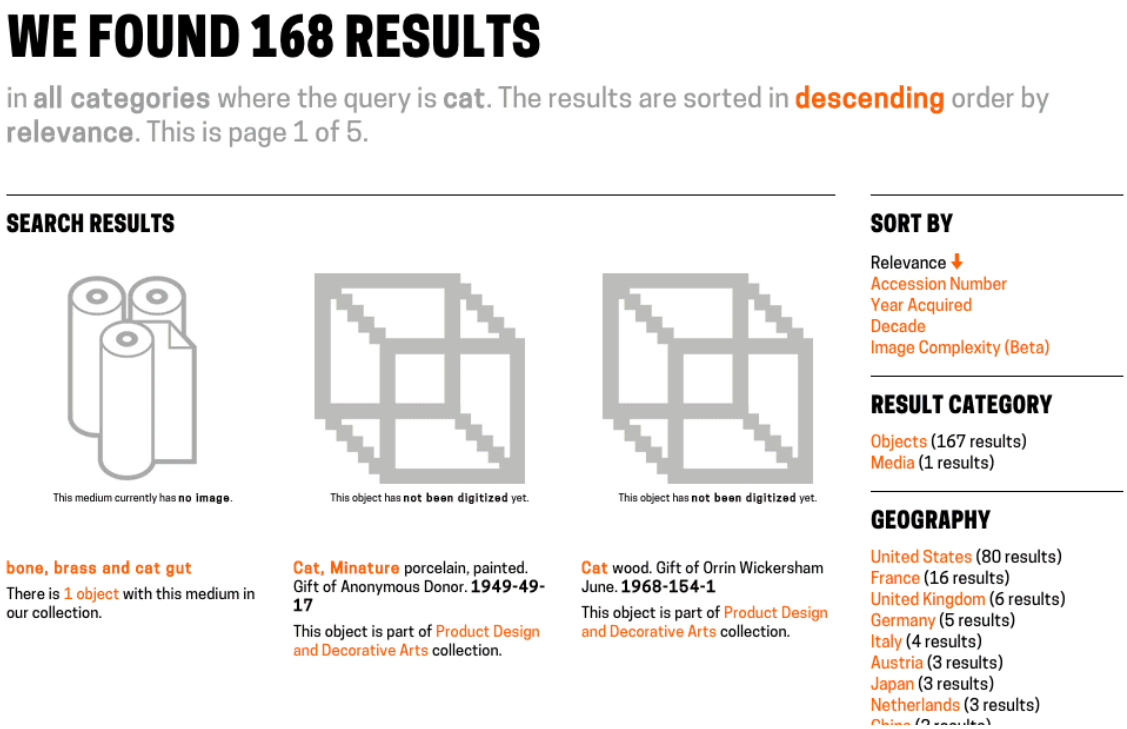
confusion as to what “fancy search” did as the search categories are now clearly labeled.



Showing the tabbed view for search options

The next challenge was how to display sorting. Previously, the drop-down menu containing sort options was hidden in a “filter these results” collapsible menu. I wanted to lay out all of the sorting options for the user to see at a glance and easily switch between sorting modes. Instead of placing them across the top in a container that would push the search results further down the page, I moved them to a sidebar which would also house search result facets (more on that soon). While it does cut in to our ability to display the pictures as big as we’d

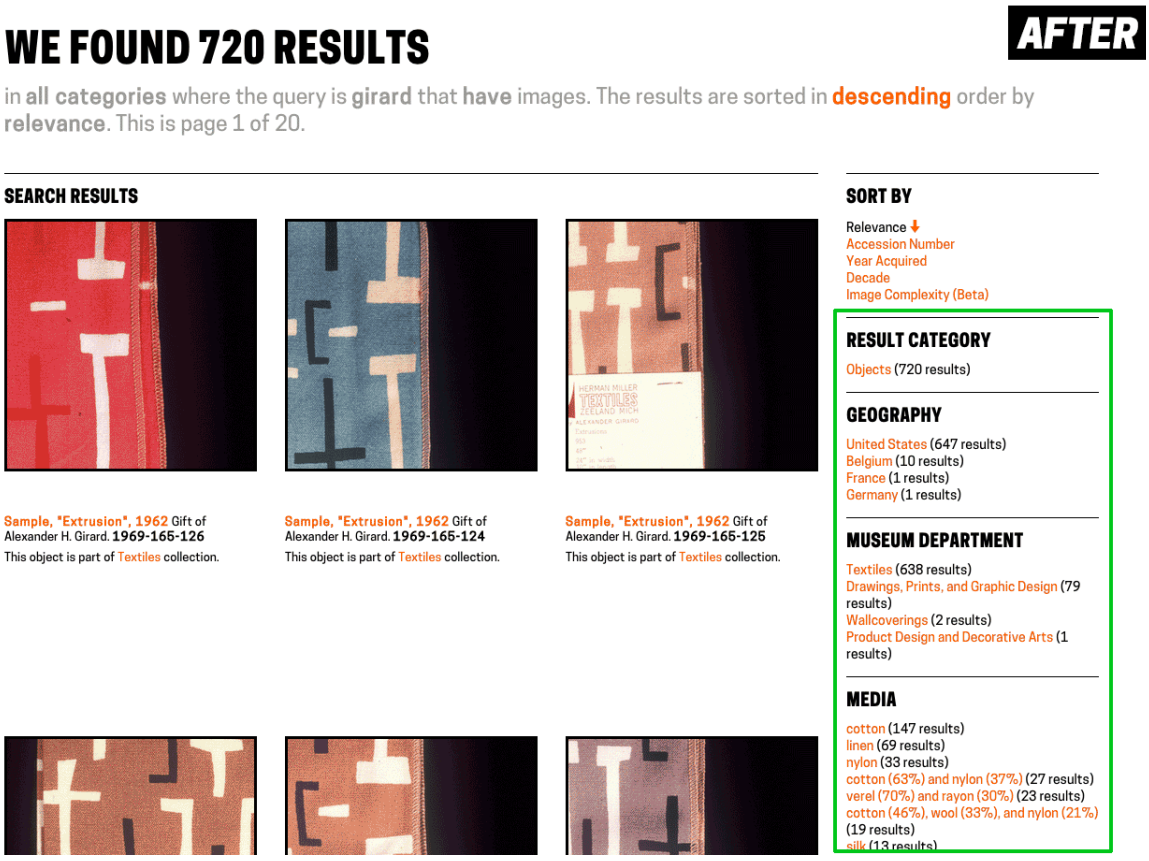
like, it's the only way we can avoid hiding information from the user. Placing these options in a collapsible menu creates two problems: if the menu is collapsed by default, we're basically ensuring that nobody will ever use them. If the menu is expanded by default, then it means that the actual results are no longer the most important thing on the page (which, on a search results page, they clearly are). The sidebar gives us room to lay out a lot of options in an unobtrusive but easily-accessible way[2] #f2.



Switching between sort mode and sort order.

The final challenge on the frontend was how to handle faceting. Faceting is a great way for users who know what they're looking for to narrow down options, and a great way for users who don't know what they're looking for to be exposed to the various buckets we're able to place objects in to.

Previously on our frontend, faceting was only available on fancy search. We displayed a few of the faceted fields across the top of the results page, and if you wanted further control, users could select individual fields to facet on using a drop-down menu at the bottom of the fancy search form. When they used this, though, the results page displayed only the facets, not the objects. In my updates, I've turned faceting on for all searches. They appear alongside the search results in the sidebar.



Relocating facets from across the top of the page to the sidebar

Doing it Live

We initially rolled these changes out about 10 days ago, though they were hidden from users who didn't know the URL. This was to prove to ourselves that we could run Elasticsearch and Solr alongside each other without the whole site blowing up. We're still using Solr for a bit more than just the search (for example, to show which people have worked with a given person `https://collection.cooperhewitt.org/people/18043523/collaborators/producers/`), so until we migrate completely to Elasticsearch, we need to have both running in parallel.

A few days later, I flipped the switch to make Elasticsearch the default search provider and passed the link around internally to get some feedback from the rest of the museum. The feedback I got was important not just for working out the initial bugs and kinks, but also (and especially for myself as a relative newbie to the museum world) to help me get the language right and consider all the different expectations users might have when searching our collection. This resulted in some tweaks to the layout and copy, and some added functionality, but

mostly it will inform my bigger-picture design decisions going forward.

A Few Numbers...

Improving performance wasn't a primary objective in our changes to search, but we got some speed boosts nonetheless.

QUERY	BEFORE (SOLR)	AFTER (ELASTICSEARCH)
query=cat, facets on	162 results in 1240-1350ms https://collection.cooperhewitt.org/search/fancy?q=cat	167 results in 450-500ms https://collection.cooperhewitt.org/search/collection/?query=cat
year_acquired=gt1990, facets on	13,850 results in 1430-1560ms https://collection.cooperhewitt.org/search/fancy?y=gt1990	14,369 results in 870-880ms https://collection.cooperhewitt.org/search/collection/?year_acquired=gt1990
department_id=35347493&period_id=35417101, facets on	1,094 results in 1530-1580ms https://collection.cooperhewitt.org/search/fancy?D=35347493&PR=35417101	1,150 results in 960-990ms https://collection.cooperhewitt.org/search/collection/?department_id=35347493&period_id=35417101

There are also cases where queries that turned up nothing before now produce relevant results, like "red concert poster, https://collection.cooperhewitt.org/search/collection/?query=red%20concert%20poster" (0 -> 11 results) "German drawings https://collection.cooperhewitt.org/search/collection/?query=German+drawings" (0 -> 101 results) and "checkered Girard samples https://collection.cooperhewitt.org/search/collection/?query=checkered+girard+samples" (0 -> 10 results).

Next Steps

Getting the improved search in front of users is the top priority now – that means you! We're very interested in hearing about any issues, suggestions or general feedback that you might have — leave them in the comments or tweet us @cooperhewittlab https://twitter.com/cooperhewittlab.

I'm also excited about integrating some more exiting search features — things like type-ahead search and related search suggestion — on to the site in the future. Additionally, figuring out how to let users make super-specific queries (like the aforementioned "most common medi-

um in objects between 1990-2000") is a challenge that will require a lot of experimentation and testing, but it's definitely an ability we want to put in the hands of our users in the future.

New Search is live on our site right now – go check it out http://collection.cooperhewitt.org/search/collection!

[1] We've been struggling to find a word to use for things that are "first-class" in our collection (objects, people, countries, media etc.) that makes sense to both museum-folk and the laypeople. We can't use "objects" because those already refer to a thing that might go on display in the museum. We've also tried "items," "types" and "isas" (as in, "what is this? it is a person"). But nothing seems to fit the bill.

[2] We're not in complete agreement here at the labs over the use of a sidebar to solve this design problem, but we're going to leave it on for a while and see how it fares with time. Feedback is requested!

This entry was posted in Backends http://labs.cooperhewitt.org/category/backends/, *Experimental* http://labs.cooperhewitt.org/category/experimental/, *Interaction Design* http://labs.cooperhewitt.org/category/interaction-design/ *and tagged Larry Wall* http://labs.cooperhewitt.org/tag/larrywall/ *on August 21, 2014* [http://labs.cooperhewitt.org/2014/rethinking-search-on-the-collections-site/] *by Sam Brenner* http://labs.cooperhewitt.org/author/brenners/.

Why are we collecting source code?



http://3ug4ii3uortgp68ee31cybxjyz.wpengine.netdna-cdn.com/wp-content/uploads/2014/08/tumblr_n7nns1po1b1tddjl4o1_500.gif

(via http://rekall.tumblr.com/post/92033337743)

Part of what we continue to work on in parallel to the opening of Cooper Hewitt http://www.cooper-hewitt.org/new-experience/ is capacity building for the museum to collect 'the present' – which includes the code that underpins and makes functional much of the 'designs' of the modern world. That means all the interactive, networked design 'objects/works', not just on screens but also those embedded in products, services and systems. I'm not just interested in this for 'digital preservation' reasons, but also to help us come up with new ways to interpret, contextualise and communicate the 'how and why' of these objects (and the choices the designers made) to our visitors.

Aaron liked what I wrote to a designer with whom we are working with on collecting some interactive pieces, and thought it made sense to share it in a redacted form. Sometimes it is nice to be asked to be explicit about why the underlying code matters – and so here's what I wrote.

As the (publicly-funded) national design museum, one of the reasons we are interested in acquiring the underlying code and data is that allows the museum and future scholars and researches to explicitly explore and interrogate the choices and decisions made at the time of a work's creation in response the the technological constraints of the time, as well as the adjustments made through a work's creation to make it better respond to the needs of users. In the case of Planetary http://www.cooperhewitt.org/2013/08/26/planetary-collecting-and-preserving-code-as-a-living-object/ *this is why we acquired the entire Github repository* https://github.com/cooperhewitt/planetary *– the versioned codebase.*

Approaching your choices of language and platform as 'materials' that were shaped by the period in which the work was made, as well as your decisions in the code itself, is extremely useful for interpretation and future scholarship. Nick Monfort & Ian Bogost's book on the affordances of the Atari 2600 platform, Racing the Beam http://mitpress.mit.edu/books/racing-beam*, is just one example of the kind of scholarship we foresee as being possible when code and data is acquired with works. This sort of exploration – of decisions made, and the technological and social constraints encountered – is key to Cooper Hewitt helping the public to interrogate and understand works in the collection and the work of designers as more than just aesthetic experiences.*

Increasingly when we are acquiring interactive works we are also interested in how users used and reacted to them. In these cases we would also consider acquiring user research, user feedback and usage data along with a work – so that future scholars and visitors could understand a work's success in achieving its stated goals. In terms of product design collections this is often reduced to 'market and sales performance' but we feel that in the case of works on the internet there is far more potential opportunity to explore other more complex and nuanced measures of relative 'success' that reveal the work that interaction designers and the choices they make.

In respect to [redacted] specifically, it helps visitors understand that you made this work in a particular way when you did because that's how the technology and access to data was at the time. And that if that it was to remade now in 2014, there might be a multiplicity of new ways to do it now and we can explicitly talk about the differences.

The other reason is that the underlying code and data better enables the museum to preserve these works as part of the Smithsonian's collection indefinitely in the public trust – and perhaps exhibit them 100 years from now.

Discuss.

This entry was posted in Meta Issues http://labs.cooperhewitt.org/category/meta-issues/ *on August 25, 2014* [http://labs.cooperhewitt.org/2014/why-are-we-collecting-source-code/] *by Seb Chan* http://labs.cooperhewitt.org/author/seb/.

A colophon for bias

> *The term [colophon] derives from tablet inscriptions appended by a scribe to the end of a … text such as a chapter, book, manuscript, or record. In the ancient Near East, scribes typically recorded information on clay tablets. The colophon usually contained facts relative to the text such as associated person(s) (e.g., the scribe, owner, or commissioner of the tablet), literary contents (e.g., a title, "catch" phrase, number of lines), and occasion or purpose of writing.*
>
> — *Wikipedia* http://en.wikipedia.org/wiki/Colophon_%28publishing%29

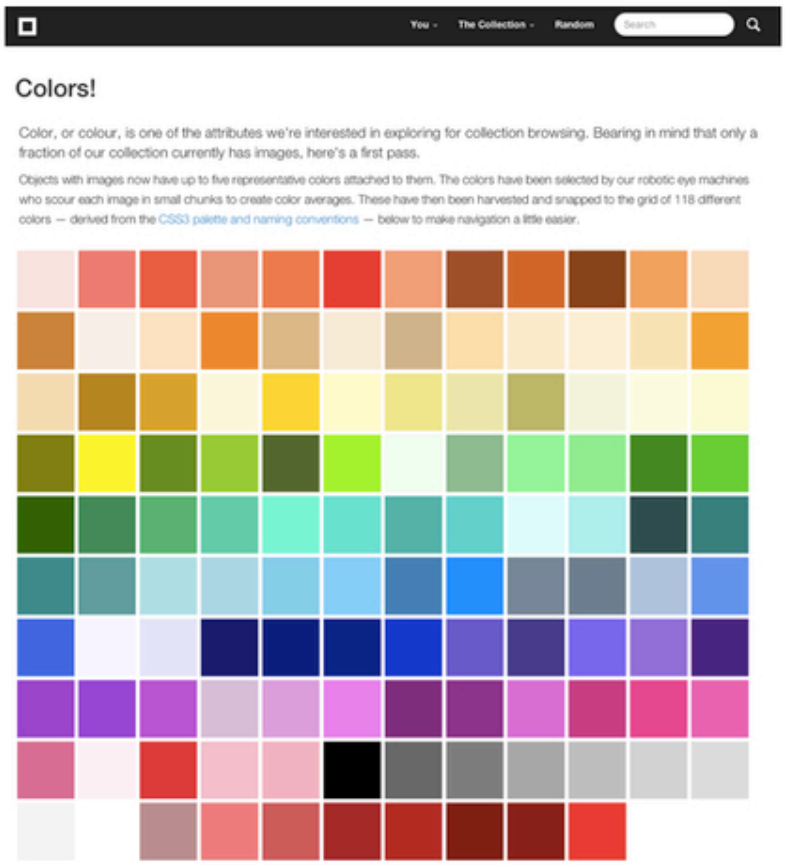
A couple of months ago we added the ability to search the collections website by color http://www.collection.cooperhewitt.org/objects/colors using more than one palette http://collection.cooperhewitt.org/objects/colors/palettes/. A brief refresher: Our search by color functionality works http://labs.cooperhewitt.org/2013/giv-do/ by first extracting the dominant palette for an index. That means the top 5 colors out of a possible 32 million choices. 32 million is too large a surface area to search against so each of the five results are then "snapped" to their closest match on a much smaller grid of possible colors. These matches are then indexed and used to query our database when someone searches for objects matching a specific color.

It turns out that the CSS3 color palette https://developer.mozilla.org/en-US/docs/Web/CSS which defines a fixed set of 138 colors is an excellent choice for doing this sort of thing. CSS is the acronym for Cascading Style Sheets (CSS) which is a "language used to describe the presentation" of a webpage separate from its content. Instead of asking people searching the collections website to be hyper-specific in their queries we take the color they are searching for and look for the nearest match in the CSS palette.

For example: #ef0403 https://collection.cooperhewitt.org/objects/colors/css3/ef0403/ becomes #ff0000 https://collection.cooperhewitt.org/objects/colors/css3/ff0000/ or "red". #f2e463 https://collection.cooperhewitt.org/objects/colors/css3/f2e463/ becomes #f0e68c https://collection.cooperhewitt.org/objects/colors/css3/f0e68c/ or "khaki" and so on.

This approach allows us to not only return matches for a specific color but also to show ob-

jects that are more *like* a color than not. It's a nice way to demonstrate the breadth of the collection and also an invitation to pair objects that might never be seen together.



http://collection.cooperhewitt.org/objects/colors/

From the beginning we've always planned to support multiple color palettes. Since the initial search-by-color functionality was built in a hurry with a focus on seeing whether we could get it to work *at all* adding support for multiple palettes was always going to require some re-jiggering of the original code. Which of course means that finding the time to make those changes had to compete with the crush of everything else http://www.cooperhewitt.org/new-experience/ and on most days it got left behind.

Earlier this year Rebecca Alison Meyer the 6-year old daughter http://meyerweb.com/eric/thoughts/category/personal/rebecca/ of Eric Meyer, a long-standing member of the CSS community, died of cancer. Eric's contributions and work to promote the CSS standard can not be overstated. http://meyerweb.com/eric/css/ The web would be an entirely other (an entirely poorer) space without his efforts and so some people suggested that a 139th color be added to the CSS Color module to rec-

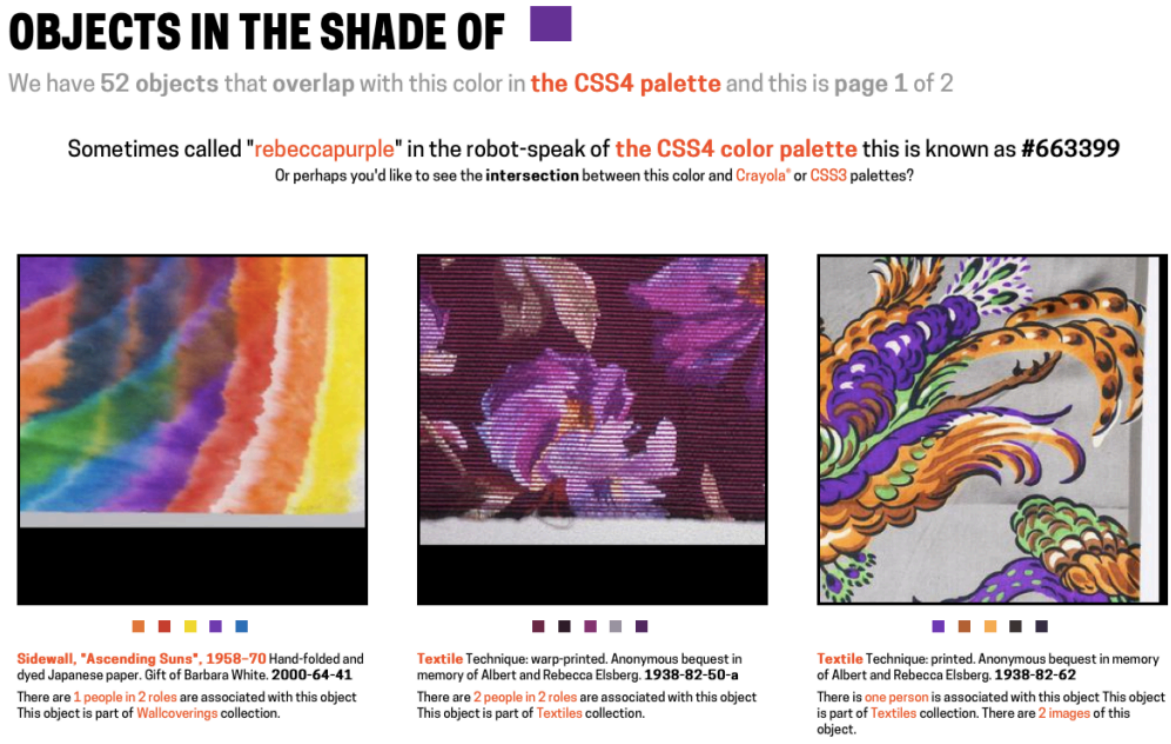
ognize his work and honor his daughter. In June Dominique Hazaël-Massieux wrote http://dis-course.specifiction.org/t/name-663399-becca-purple-in-css4-color/225:

> *I'm not sure about how one goes adding names to CSS colors, and what the specific purpose they fulfill, but I think it would be a good recognition of @meyerweb http://www.twitter.-com/meyerweb/ 's impact on CSS, and a way to recognize that standardization is first and foremost a social process, to name #663399 color "Becca Purple".*

In reply Eric Meyer wrote http://meyerweb.com/eric/thoughts/2014/06/19/rebeccapurple/:

> *I have been made aware of the proposal to add the named color beccapurple (equivalent to #663399) to the CSS specification, and also of the debate that surrounds it.*
>
> *I understand the arguments both for and against the proposal, but obviously I am too close to both the subject and the situation to be able to judge for myself. Accordingly, I let the editors of the Colors specification know that I will accept whatever the Working Group decides on this issue, pro or con. The WG is debating the matter now.*
>
> *I did set one condition: that if the proposal is accepted, the official name be rebeccapurple. A couple of weeks before she died, Rebecca informed us that she was about to be a big girl of six years old, and Becca was a baby name. Once she turned six, she wanted everyone (not just me) to call her Rebecca, not Becca.*
>
> *She made it to six. For almost twelve hours, she was six. So Rebecca it is and must be.*

Shortly after that #663399 http://dev.w3.org/csswg/css-color/#valdef-color-rebeccapurple or rebeccapurple http://dev.w3.org/csswg/css-color/#valdef-color-rebeccapurple was added to the CSS4 Colors module specification. At which point it only seemed right to finally add support for multiple color palettes to the collections website.



https://collection.cooperhewitt.org/objects/colors/css4/663399

Over the course of a month or so, in the margins of day, all of the search-by-color code was rewritten to work with more than a single palette and now you can search the collection for objects in the shade of rebeccapurple https://collection.cooperhewitt.org/objects/colors/css4/663399/.

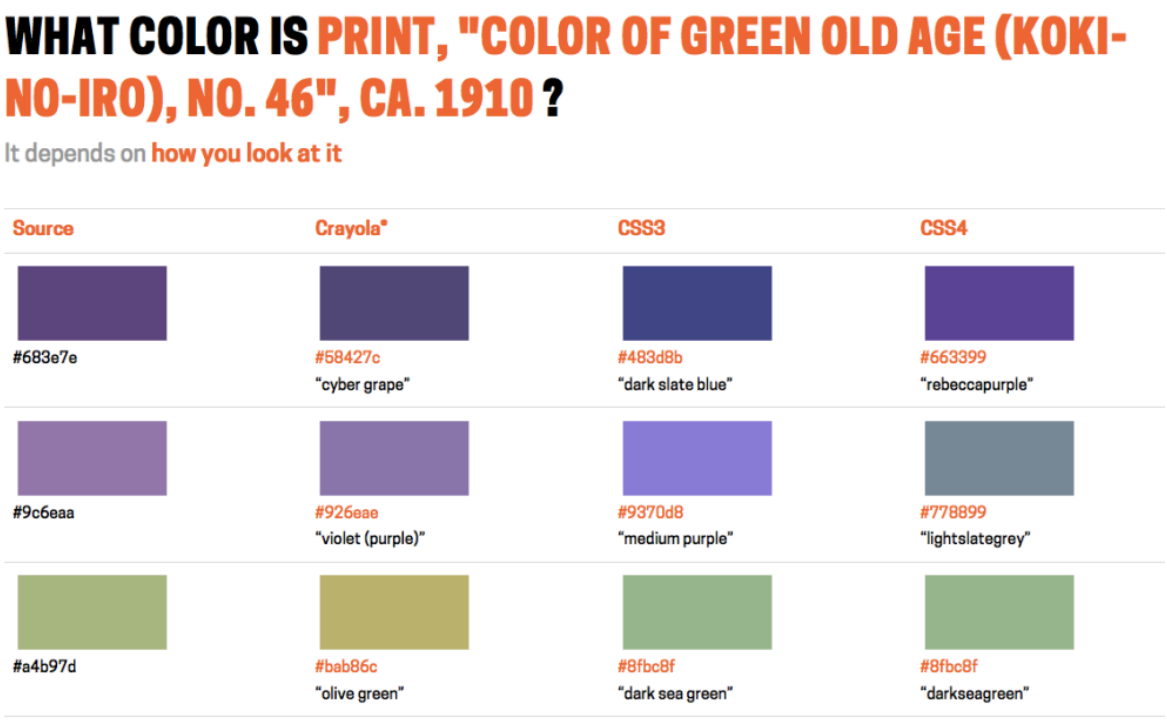
In addition to the CSS3 https://collection.cooperhewitt.org/objects/colors/palettes/css3/ and CSS4 https://collection.cooperhewitt.org/objects/colors/palettes/css4/ color palettes we also added support for the Crayola https://collection.cooperhewitt.org/objects/colors/palettes/crayola/ color palette. For example, the closest color to "rebeccapurple" https://collection.cooperhewitt.org/objects/colors/css4/663399/ in the Crayola scheme of things is "cyber grape" https://collection.cooperhewitt.org/objects/colors/crayola/663399/.

You can see all the possible nearest-colors for an object by appending `/colors` to an object page URL. For example:

https://collection.cooperhewitt.org/objects/18380795**/colors** https://collection.cooperhewitt.org/objects/18380795/colors

The dominant color for this object is #683e7e https://collection.cooperhewitt.org/objects/colors/683e7e/ which maps to #58427c https://collection.cooperhewitt.org/objects/colors/crayola/58427c or “cyber grape” in Crayola-speak and #483d8b https://collection.cooperhewitt.org/objects/colors/css3/483d8b or “dark slate blue” in CSS3-speak and #663399 https://collection.cooperhewitt.org/objects/colors/css4/663399 or “rebeccapurple” in CSS4-speak.

Now that we’ve done the work to support multiple palettes the only limits to adding more is time and imagination. I would like to add a greyscale palette. I would like to add one or more color-blind palettes. I would especially like to add a “blue” palette – one that spans non-photo blue https://en.wikipedia.org/wiki/Non-photo_blue through International Klein Blue https://en.wikipedia.org/wiki/International_Klein_Blue all the way to Kind of Bloop midnight blue http://creativemornings.com/talks/andy-baio/2?play=0 just to see where along that spectrum objects which aren’t even a little bit blue would fall.



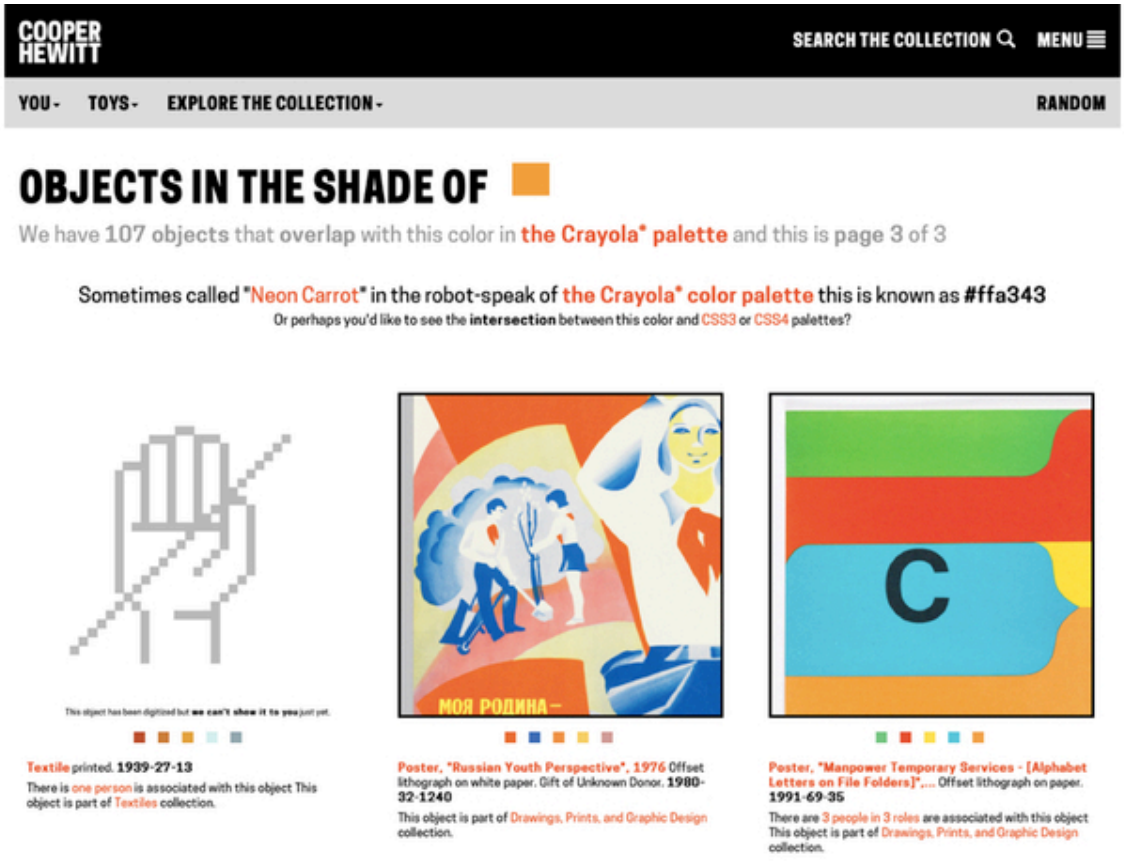
https://collection.cooperhewitt.org/objects/18380795/colors

The point being that there are any number of color palettes that we can devise and use as a lens through which to see our collection. Part of the reason we chose to include the Crayola color palette in version “2” of search-by-color is because the colors they’ve chosen have been given expressive names whose meaning is richer than the sum of their descriptive parts. What does it mean for an object’s colors to be described as macaroni and cheese https://collec-

tion.cooperhewitt.org/objects/colors/crayola/ffbd88/ -ish or outer space https://collection.cooperhewitt.org/objects/colors/crayola/414a4c/ -ish in nature? Erika Hall's 2007 talk Copy is Interface http://www.slideshare.net/mulegirl/copy-is-interface is an excellent discussion of this idea.

I spoke about some of these things http://www.aaronland.info/weblog/2014/09/12/colophon/#debate last month at the The Search is Over http://www.searchisover.org/ workshop, in London. I described the work we have done on the collections website, to date, as a kind of managing of absence http://labs.cooperhewitt.org/2013/albers-boxes/ . Specifically the absence of metadata and ways to compensate for its lack or incompleteness while still providing a meaningful catalog and resource.

It is through this work that we started to articulate the idea that: The value of the whole in aggregate, for all its flaws, outweighs the value of a perfect subset. The irregular nature of our collection metadata http://www.github.com/cooperhewitt/collection/ has also forced us to consider that even if there were a single unified interface to convey the complexities of our collection it is not a luxury we will enjoy any time soon.



https://collection.cooperhewitt.org/objects/colors/crayola/ffa343/

Further the efforts of more and more institutions (the Cooper Hewitt included) to embark on mass-digitization projects forces an issue that we, as a sector, have been able to side-step until now: That no one, including lots of people who actually work at museums, have ever seen much of the work in our collections. So in relatively short-order we will transition from a space defined by an absence of data to one defined by a surfeit of, at the very lest, photographic evidence that no one will know how to navigate.

To be clear: This is a *good problem* to have but it does mean that we will need to starting thinking about models to recognize the shape of the proverbial elephant in the room http://www.engineering.com/DesignerEdge/DesignerEdgeArticles/ArticleID/6302/Billy-the-LiDAR-Elephant.aspx and building tools to see it.

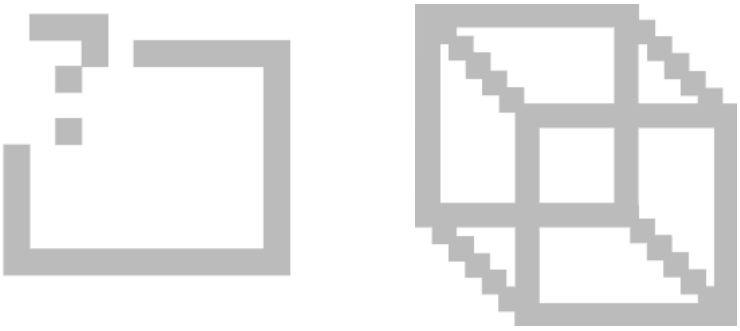
It is in those tools that another equally important challenge lies. The scale and the volume of the mass-digitization projects being undertaken means that out of necessity any kind of first-pass cataloging of that data will be done by machines. There simply isn't the time (read: money) to allow things to be cataloged by human hands and so we will inevitably defer to the opinion of computer algorithms.

This is not necessary as dour a prediction as it might sound. Color search http://collection.cooperhewitt.org/objects/colors/ is an example of this scenario and so far it's worked out pretty well for us. What search-by-color and other algorithmic cataloging http://goodformandspectacle.wordpress.com/2014/10/23/internal-rd-project-1-netflix-o-matic/ points to is the need to develop an iconography, or a colophon, to indicate machine bias. To design and create language and conventions that convey the properties of the "extruder" that a dataset has been shaped by.



http://3ug4ii3uortgp68ee31cybxjyz.wpengine.netdna-cdn.com/wp-content/uploads/2014/10/search-is-over.033-640.jpg

Those conventions don't really exist yet. Bracketing search by color with an identifiable palette (a bias) is one stab at the problem but there are so many more places where we will need to signal the meaning (the subtext?) of an automated decision. We've tried to address one facet of this problem with the different graphic elements we use to indiciate the reasons why an object may not have an image http://www.aaronland.info/weblog/2013/05/05/design-thanking/#face .





Left to right: We're supposed to have a picture for this object… but we can't find it; This object has not been photographed; This object has been photographed but for some reason we're not allowed to show it to you… you know, even though it's been acquired by the Smithsonian.

Another obvious and (maybe?) easy place to try out this idea is search http://collection.cooperhewitt.org/search itself. Search engines are not, in fact, magic. Most search engines work the same way: A given string is "tokenized" and then each resultant piece is "filtered". For the example the phrase "checkered Girard samples https://collection.cooperhewitt.org/search/collection/?query=checkered+girard+samples" might typically be tokenized by splitting things on whitespace but you could just as easily tokenize it by any pattern that can be expressed to a computer https://xkcd.com/208/. So depending on your tokenized you might end up with a list like:

- checkered
- Girard
- samples

Or:

- checkered Girard
- samples

Each one of those "tokens" are then analyzed and filtered according to their properties. Maybe they get grouped by their phonetics https://en.wikipedia.org/wiki/Soundex, which is essentially how the snap-to-grid trick works for the collection's color search. Maybe they are grouped by what type of word they are: proper nouns, verbs, prepositions and so on. I've never actually seen a search engine that does this but there is nothing technically to prevent someone from doing it either.

The simplest and dumbest thing would be to indicate on a search results page that your query results were generated using one or more tokenizers or filters. In our case that would be (1) tokenizer and (5) filters.

Tokenizers:

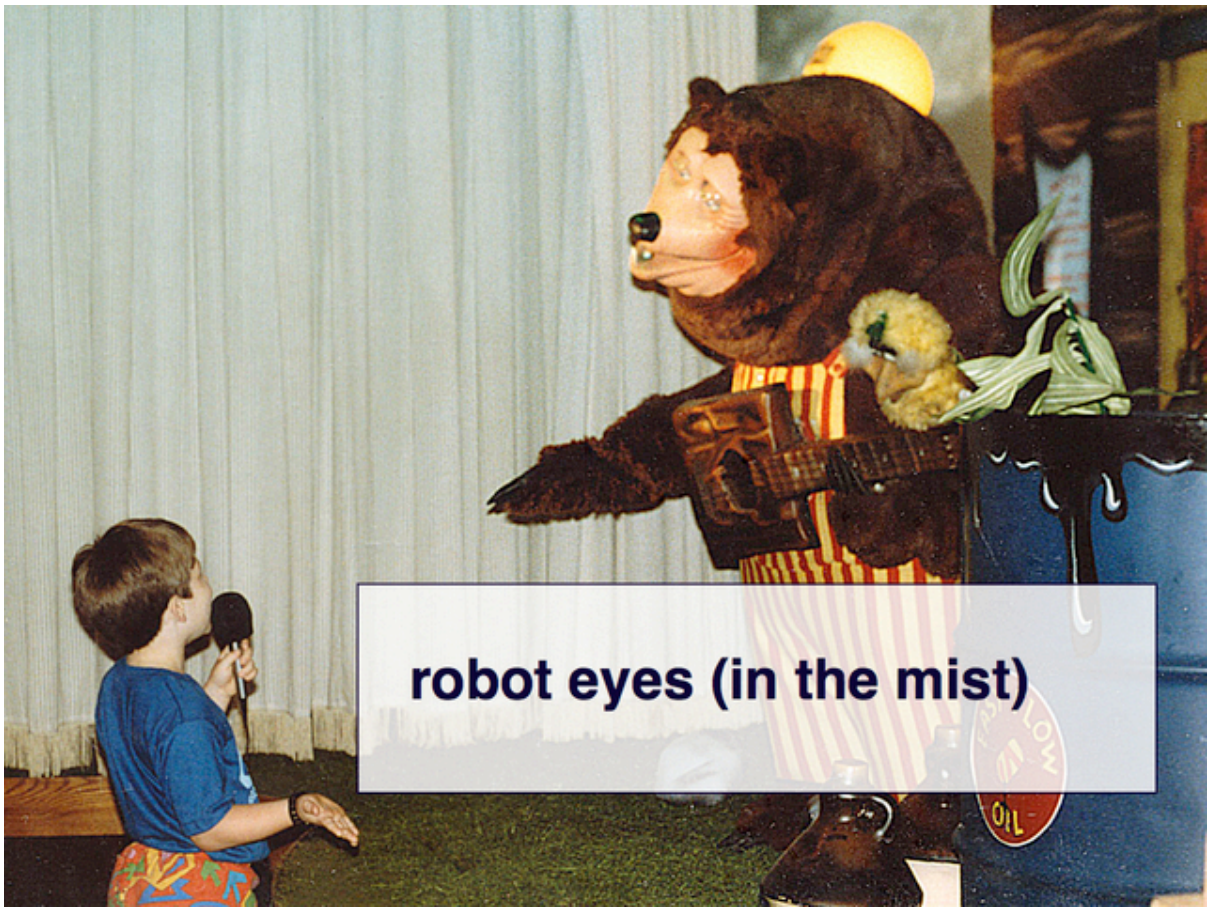
1. Unicode Standard Annex #29 http://unicode.org/reports/tr29/

Filters:

1. Remove English possessives https://lucene.apache.org/core/4_0_0/analyzers-common/org/apache/lucene/analysis/en/EnglishPossessiveFilter.html
2. Lowercase all tokens http://www.elasticsearch.org/guide/en/elasticsearch/reference/current/analysis-lowercase-tokenfilter.html
3. Ingore a set list of stopwords http://www.elasticsearch.org/guide/en/elasticsearch/reference/1.4/analysis-stop-tokenfilter.html
4. Stem tokens according to the Porter Stemming Algorithm http://snowball.tartarus.org/algorithms/porter/stemmer.html
5. Convert non-ascii characters to ascii http://www.elasticsearch.org/guide/en/elasticsearch/guide/current/asciifolding-token-filter.html

That's not very sexy or ooh-shiny http://tvtropes.org/pmwiki/pmwiki.php/Main/AttentionDeficitOohShiny but not everything needs to be. What it does, though, is provide a measure of transparency for people to gauge the reality that any result set is the product of choices which may have little or no relationship to the question being asked or the person asking that question.

These are devices, for sure, and they are not meant to replace a more considered understanding or contemplation of a topic but they can act as an important shorthand to indicate the arc of an answer's motive.



http://3ug4ii3uortgp68ee31cybxjyz.wpengine.netdna-cdn.com/wp-content/uploads/2014/10/search-is-over.038-640.jpg

And that's just for search engines. Now imagine what happens when we all start pointing computer vision algorithms http://www.robertelliottsmith.com/?p=530 at our collections…

Update: Since publishing this blog post the nice people working on the GOV.UK http://www.gov.uk/ websites launched "info" pages https://insidegovuk.blog.gov.uk/2014/10/29/info-pages-publishing-data-about-user-needs-and-metrics/. Visitors can now append `/info` to any of the pages on the gov.uk website will and see what and who and how that part of the website is supposed to do. Writing about the project they say:

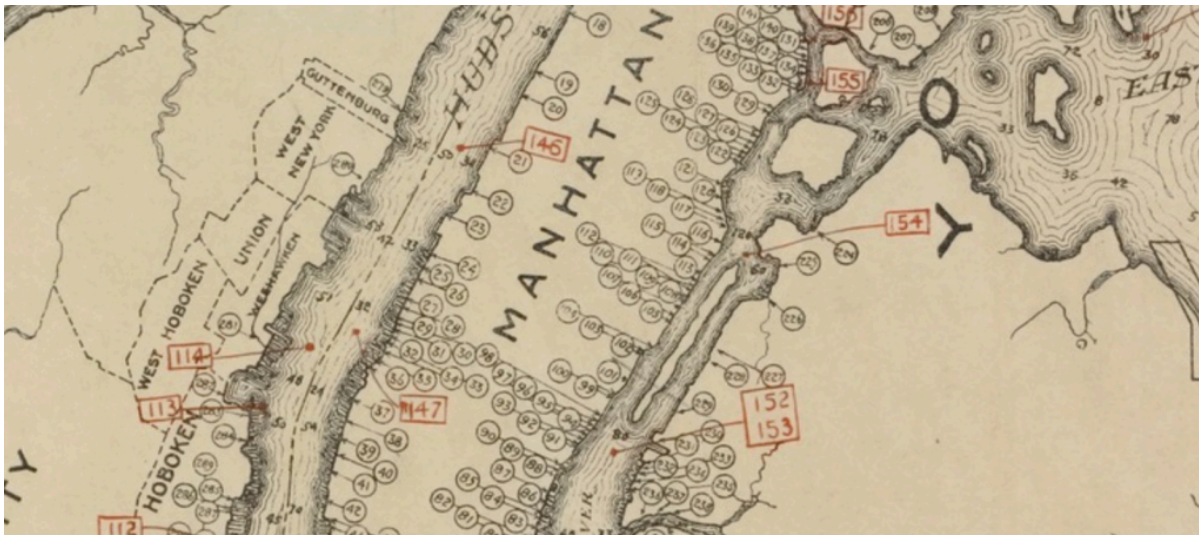
> *An 'info' page contains the user needs the page is intended to meet … Providing an easy way to jump from content to the underpinning needs allows content designers coming to a new topic to understand the need and build empathy with the users quicker. Publishing the GOV.UK user needs should also make the team's work more transparent and trace-*

able.

Bravo!

This entry was posted in Collection data http://labs.cooperhewitt.org/category/collection-data/*, Meta Issues* http://labs.cooperhewitt.org/category/meta-issues/ *on October 27, 2014* [http://labs.cooperhewitt.org/2014/a-colophon-for-bias/] *by asc* http://labs.cooperhewitt.org/author/asc/*.*

The API at the center of the museum



http://3ug4ii3uortgp68ee31cybxjyz.wpengine.netdna-cdn.com/wp-content/uploads/2014/11/1905-nyc-sewer-outlets.png

Extract from "Outline map of New York Harbor & vicinity : showing main tidal flow, sewer outlets, shellfish beds & analysis points.", New York Bay Pollution Commission, 1905. From New York Public Library http://digitalcollections.nypl.org/items/5fd484ac-dfd8-31b5-e040-e00a18060c12 .

Beneath our cities lies vast, labyrinthine sewer systems. These have been key infrastructures allowing our cities to grow larger, grow more densely, and stay healthy. Yet, save for passing interests in Urban Exploration (UrbEx https://www.flickr.com/groups/urbex/), we barely think of them as 'beautifully designed systems'. In their time, the original sewer systems were critical long term projects that greatly bettered cities and the societies they supported.

In some ways what the Labs has been working on over the past few years has been a similar infrastructure and engineering project which will hopefully be transformative and enabling for our institution as a whole. As SFMOMA's recent post http://www.sfmoma.org/about/research_projects/lab/philosophies_of_online_collections , which included an interview with Labs' Head of Engineering, Aaron Cope, makes clear, our API and the collection site that it is built upon, is a carrier for a new type of institutional philosophy.

Underneath all our new shiny digital experiences – the Pen, the Immersion Room, and other digital experiences – as well as the refreshed 'services layer' of ticketing, Pen checkouts, and object label management, lies our API. There's no readymade headline or Webby award await-

ing a beautifully designed API – and probably there shouldn't be. These things should just work and provide the benefit to their hosts that they promised.

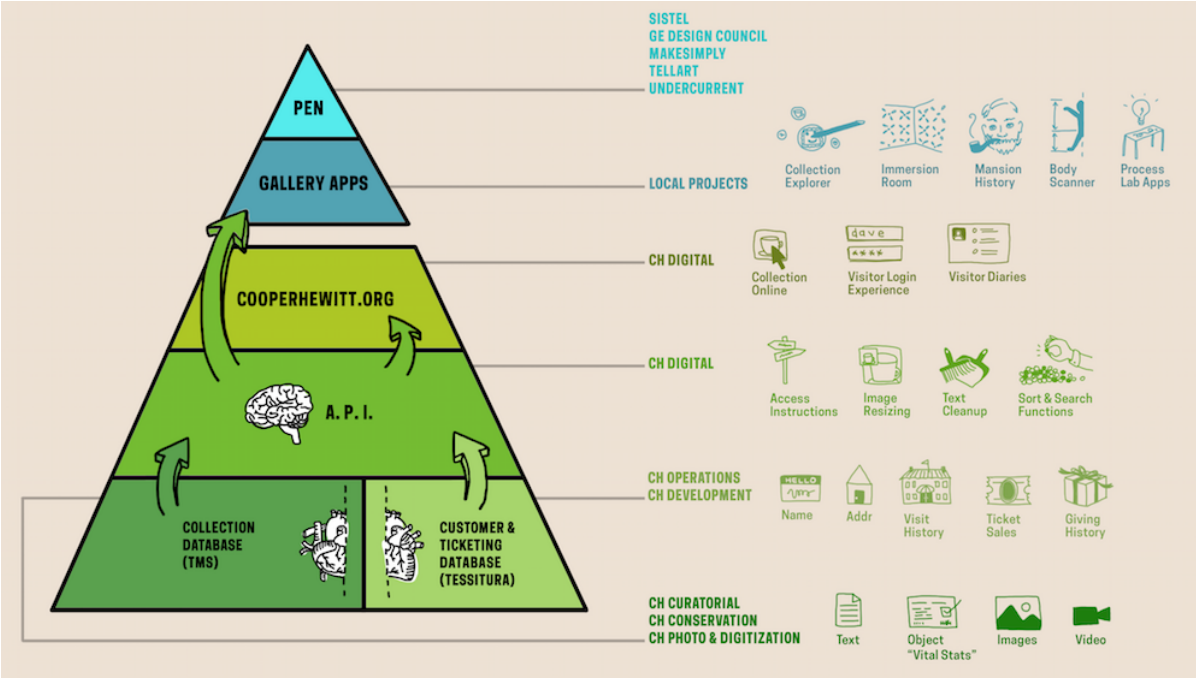
So why would a museum burden itself with making an API to underpin all its interactive experiences – not just online but in-gallery too?

Its about sustainability. Sustainability of content, sustainability of the experiences themselves, and also, importantly, a sustainability of 'process'. A new process whereby ideas can be tested and prototyped as 'actual things' written in code. In short, as Larry Wall http://en.wikipedia.org/wiki/Larry_Wall said its about making "easy things easy and hard things possible".

The overhead it creates in the short term is more than made up for in future savings. Where it might be prudent to take short cuts and create a separate database here, a black box content library there, the fallout would be unchanging future experiences unable to be expanded upon, or, critically, rebuilt and redesigned by internal staff.

Back at my former museum, then Powerhouse web manager Luke Dearnley, wrote an important paper http://www.museumsandtheweb.com/mw2011/papers/reprogramming_the_museum on the reasons to make your API central to your museum back in 2011. There the API was used internally to do everything relating to the collection online but it only had minor impact on the exhibition floor. Now at Cooper Hewitt the API and exhibition galleries are tightly intertwined. As a result there's a definite 'API tax' that is being imposed on our exhibition media partners – Local Projects and Tellart especially – but we believe it is worth it.

So here's a very high level view of 'the stack' drawn by Labs' Media Technologist, Katie https://labs.cooperhewitt.org/author/katieshelly/ .



http://3ug4ii3uortgp68ee31cybxjyz.wpengine.netdna-cdn.com/wp-content/uploads/2014/11/API_stack.png

Click http://3ug4ii3uortgp68ee31cybxjyz.wpengine.netdna-cdn.com/wp-content/uploads/2014/11/API_stack.png *to enlarge*

At the bottom of the pyramid are the two 'sources of truth'. Firstly, the collection management system into which is fed curatorial knowledge, provenance research, object labels and interpretation, public locations of objects in the galleries, and all the digitised media associated with objects, donors and people associated with the collection. There's also now the other fundamental element – visitor data. Stored securely, Tessitura operates as a ticketing system for the museum and in the case of the API operates as an identity-provider where needed to allow for personalisation.

The next layer up is the API which operates as a transport between the web and both the collection and Tessitura. It also enables a set of other functions – data cleanup and programmatic enhancement.

Most regular readers have already seen the API https://collection.cooperhewitt.org/api/ – apart from TMS, the Collection Management System, it is the oldest piece of the pyramid. It went live shortly after the first iteration of the new collections website in 2012 http://labs.cooperhewitt.org/2012/online-collection-alpha/ . But since then it has been growing with new methods added regularly. It now contains not only methods for collection access but also user authentication and account structures, and anonymised event logs. The latter of these opens up all manner of data visualization opportunities for artists and researchers down the track.

In the web layer there is the public website but also for internal museum users there are small web applications. These are built upon the API to assist with object label generation, metadata enhancement, and reporting, and there's even an aptly-named 'holodeck' for simulating all manner of Pen behaviours in the galleries.

Above this are the two public-facing gallery layers. The application and interfaces designed and built on top of the API by Local Projects, the Pen's ecosystem of hardware registration devices designed by Tellart, and then the Pen itself http://www.cooperhewitt.org/new-experience/designing-pen/ which operates as a simple user interface in its own right.

What is exciting is that all the API functionality that has been exposed to Local Projects and Tellart to build our visitor experience can also progressively be opened up to others to build upon.

Late last year students in the Interaction Design class at NYU's ITP program spent their semester building a range of weird http://kgitp.com/FITH/index.html and wonderful applications http://www.eu-nice.metartcenter.com/?p=100, games http://vimeo.com/96330470 and websites http://www.michellelin.com/227640/3199171/work/the-curator-project on top of the basic API. That same class (and the interested public in general) will have access to far more powerful functionality and features once Cooper Hewitt opens in December.

The API https://collection.cooperhewitt.org/api/ is here for you to use.

This entry was posted in CH 3.0 http://labs.cooperhewitt.org/category/ch-3-0/ *and tagged Larry Wall* http://labs.cooperhewitt.org/tag/larrywall/ *on November 7, 2014* [http://labs.cooperhewitt.org/2014/the-api-at-the-center-of-the-museum/] *by Seb Chan* http://labs.cooperhewitt.org/author/seb/.

HTTP ponies

Most of the image processing for the collections website http://collection.cooperhewitt.org/ is done using the Python http://www.python.org/ programming language. This includes things like: extracting colours http://labs.cooperhewitt.org/2013/giv-do/ or calculating an image's entropy http://labs.cooperhewitt.org/2013/default-sort-or-what-would-shannon-do/ (its "busy-ness") or generating those small halftone versions of image http://images.collection.cooperhewitt.org/23576_4d2429d0623da461_d.gif that you might see while you wait for a larger image to load.

Soon we hope to start doing some more sophisticated computer vision related work which will almost certainly mean using the OpenCV http://opencv.org/ tool chain. This likely means that we'll continue to use Python because it has easy to use and easy to install bindings http://docs.opencv.org/trunk/doc/py_tutorials/py_tutorials.html to hide most of the fiddly bits required to look at images with "robot eyes".



The collections website itself is not written in Python http://www.aaronland.info/weblog/2012/11/09/jello/#parallel-tms and that's okay. There are lots of ways for different languages to hold hands inside of a single "application" and we've used many of them. But we also think that most of these little pieces of functionality are useful in and of themselves and shouldn't require that a person (including us) have to burden themselves with the minutiae of the collections website infrastructure to use them.

We've slowly been taking the various bits of code we've written over the years and putting them in to discrete libraries that can then be wrapped up in little standalone HTTP "pony" or "plumbing" servers. This idea of exposing bespoke pieces of functionality via a web server is

hardly new. Dave Winer has been talking about "fractional horsepower HTTP servers http://scripting.com/davenet/1997/09/14/FractionalHorsepowerHTTPSe.html" since 1997. What's changed between then and now is that it's more fun to say "HTTP pony https://pinboard.in/u:straup/t:httpony" and it's much easier to bake a little web server in to an application because HTTP has become the lingua franca of the internet and that means almost every programming language in use today knows how to "speak" it.

In practice we end up with a "stack" of individual pieces that looks something like this:

1. Other people's code that usually does all the heavy-lifting. An example of this might be Giv Parvaneh's RoyGBiv http://roygbiv.givp.org/ library for extracting colours from images or Mike Migurski's Atkinson https://github.com/migurski/atkinson library for dithering images.
2. **A variety of cooperhewitt.* https://github.com/cooperhewitt/?query=py-cooperhewitt- libraries to hide the details of other people's code.**
3. **The cooperhewitt.flask.http_pony https://github.com/cooperhewitt/py-cooperhewitt-flask library which exports a setup of helper utilities for the running Flask-based HTTP servers. Things like: doing a minimum amount of sanity checking for uploads and filenames or handling (common) server configuration details.**
4. **A variety of plumbing-SOMETHING-server https://github.com/cooperhewitt/?query=plumbing- HTTP servers which export functionality via HTTP GET and POST requests. For example: plumbing-atkinson-server https://github.com/cooperhewitt/plumbing-atkinson-server, plumbing-palette-server https://github.com/cooperhewitt/plumbing-palette-server and so on.**
5. Flask http://flask.pocoo.org/, a self-described "micro-framework" which is what handles all the details of the HTTP call and response life cycle.
6. Optionally, a WSGI-compiliant server-container-thing-y https://www.python.org/dev/peps/pep-0333/ for managing requests to a number Flask instances. Personally we like gunicorn http://www.gunicorn.org but there are many to choose from.

Here is a *not-really-but-treat-it-like-pseudo-code-anyway* example without any error handling for the sake of brevity of a so-called "plumbing" server:

```
# Let's pretend this file is called 'example-server.py'.

import flask
from flask_cors import cross_origin
import cooperhewitt.example.code as code
import cooperhewitt.flask.http_pony as http_pony

app = http_pony.setup_flask_app('EXAMPLE_SERVER')

@app.route('/', methods=['GET', 'POST'])
@cross_origin(methods=['GET', 'POST'])
def do_something():

    if flask.request.method=='POST':
       path = http_pony.get_upload_path(app)
    else:
       path = http_pony.get_local_path(app)

    rsp = code.do_something(path)
    return flask.jsonify(rsp)

if __name__ == '__main__':
    http_pony.run_from_cli(app)
```

So then if we just wanted to let Flask take care of handling HTTP requests we would start the server like this:

```
$> python example-server.py -c example-server.cfg
```

And then we might talk to it like this:

```
$> curl -X POST -F 'file=@/path/to/file' http://localhost:5000
```

Or from the programming language of our choosing:

```
function example_do_something($path){
        $url = "http://localhost:5000";
        $file = curl_file_create($path);
        $body = array('file' => $file);
        $rsp = http_post($url, $body);
        return $rsp;
}
```

Notice the way that all the requests are being sent to `localhost`? We don't expose any of these servers to the public internet or even between different machines on the same network. But we like having the flexibility to do that if necessary.

Finally if we just need to do something natively or want to write a simple command-line tool we can ignore all the HTTP stuff and do this:

```
$> python
>>> import cooperhewitt.example.code as code
>>> code.do_something("/path/to/file")
```

Which is a nice separation of concerns. It doesn't mean that programs write themselves but they probably shouldn't anyway.

If you think about things in terms of bricks and mortar you start to notice that there is a bad habit in (software) engineering culture of trying to standardize the latter or to treat it as if, with enough care and forethought, it might achieve sentience.

That's a thing we try to guard against. Bricks, in all their uniformity, are awesome but the point of a brick is to enable a multiplicity of outcomes so we prefer to leave those details, and the work they entail, to people rather than software libraries.



Most of this code has been open-sourced and hiding in plain sight https://github.com/cooperhewitt/ for a while now but since we're writing a blog post about it all, here is a list of related tools and libraries. These all fall into categories 2, 3 or 4 in the list above.

- cooperhewitt.swatchbook https://github.com/cooperhewitt/py-cooperhewitt-swatchbook — Functions for working with colour palettes.
- cooperhewitt.roboteyes.atkinson https://github.com/cooperhewitt/py-cooperhewitt-roboteyes-atkinson — Functions for rendering halftone images using Bill Atkinson's dithering https://en.wikipedia.org/wiki/Bill_Atkinson technique in both pure-Python (slow) or C (fast) if the `atk` library is

available.

- cooperhewitt.roboteyes.colors https://github.com/cooperhewitt/py-cooperhewitt-roboteyes-colors — Functions for extracting colours from an image using a specific palette (as defined by the `cooperhewitt.swatchbook` library).
- cooperhewitt.roboteyes.opencv https://github.com/cooperhewitt/py-cooperhewitt-roboteyes-opencv — A variety of OpenCV http://www.opencv.org related functions. This one doesn't really do anything yet but we're including it here for good measure.
- cooperhewitt.roboteyes.shannon https://github.com/cooperhewitt/py-cooperhewitt-roboteyes-shannon — Functions for measuring an image's entropy and for calculating where to crop an image when generating thumbnails.
- cooperhewitt.roboteyes https://github.com/cooperhewitt/py-cooperhewitt-roboteyes — A meta library whose only purpose is to install all the other `py-cooperhewitt-roboteyes` libraries at once.

- cooperhewitt.flask https://github.com/cooperhewitt/py-cooperhewitt-flask — Utility functions for writing Flask-based HTTP applications. The most important thing to remember about things in this class is that they are *utility functions*. They simply wrap some of the boilerplate tasks required to set up a Flask application but you will still need to take care of all the details.

- plumbing-atkinson-server https://github.com/cooperhewitt/plumbing-atkinson-server — A simple Flask-based HTTP pony server to dither images.
- plumbing-shannon-server https://github.com/cooperhewitt/plumbing-shannon-server — A simple Flask-based HTTP pony server for extracting "Shannon-related" properties from images.
- plumbing-palette-server https://github.com/cooperhewitt/plumbing-palette-server — A simple Flask-based HTTP pony server for extracting colors from images.
- plumbing-bauta-server https://github.com/cooperhewitt/plumbing-bauta-server — A simple Flask-based HTTP pony server for doing OpenCV related processing. This one, like cooperhewitt.roboteyes.opencv https://github.com/cooperhewitt/py-cooperhewitt-roboteyes-opencv, doesn't really do anything yet but it will and I just like saying "bauta https://en.wikipedia.org/wiki/Carni-

val_of_Venice#Bauta" in the context of face detection http://cvdazzle.com/.

Everything has a standard Python `setup.py` for installing all the required bits (and more importantly dependencies) in all the right places. Hopefully this will make it easier for us break out little bits of awesomeness as free agents and share them with the world. The proof, as always, will be in the doing.



We've also released go-ucd https://github.com/cooperhewitt/go-ucd which is a set of libraries and tools written in Go http://golang.org/ for working with Unicode data. Or more specifically, for the time being since they are not general purpose Unicode tools, looking up the corresponding ASCII

name for a Unicode character.

For example:

```
$> ucd 罠
NET; WEB; NETWORK, NET FOR CATCHING RABBIT
```

Or:

```
$> ucd THIS → WAY
LATIN CAPITAL LETTER T
LATIN CAPITAL LETTER H
LATIN CAPITAL LETTER I
LATIN CAPITAL LETTER S
SPACE
RIGHTWARDS ARROW
SPACE
LATIN CAPITAL LETTER W
LATIN CAPITAL LETTER A
LATIN CAPITAL LETTER Y
```

There is, of course, a handy “pony” server (called `ucd-server`) for asking these questions over HTTP:

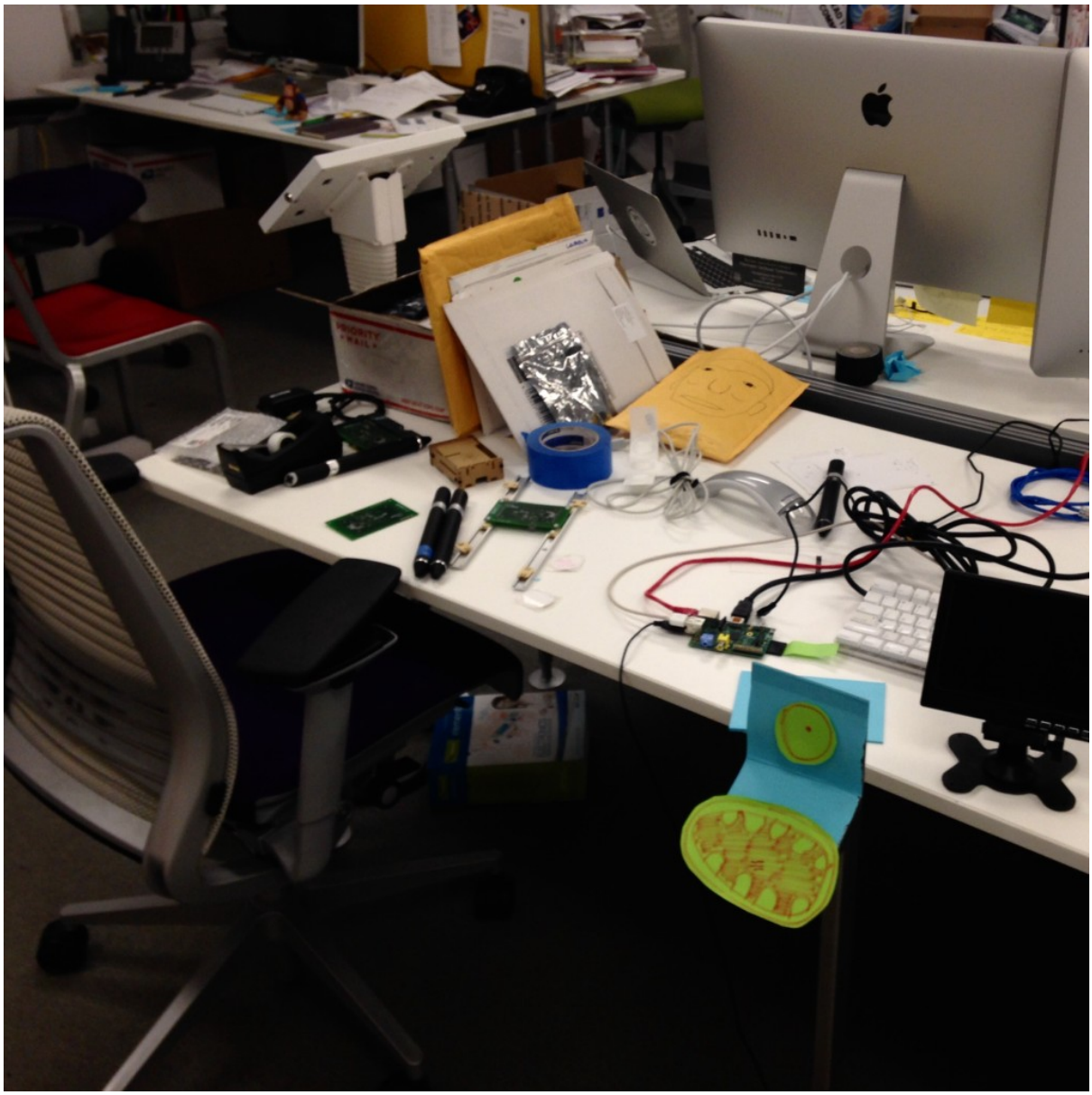
```
$> curl -X GET -s 'http://localhost:8080/?text=♕%20HAT' | python -mjson.tool
{
    "Chars": [
        {
            "Char": "\u2655",
            "Hex": "2655",
            "Name": "WHITE CHESS QUEEN"
        },
        {
```

```
            "Char": " ",
            "Hex": "0020",
            "Name": "SPACE"
        },
        {
            "Char": "H",
            "Hex": "0048",
            "Name": "LATIN CAPITAL LETTER H"
        },
        {
            "Char": "A",
            "Hex": "0041",
            "Name": "LATIN CAPITAL LETTER A"
        },
        {
            "Char": "T",
            "Hex": "0054",
            "Name": "LATIN CAPITAL LETTER T"
        }
    ]
}
```


This one, potentially, has a very real and practical use-case but it's not something we're quite ready to talk about yet. In the meantime, it's a fun and hopefully useful tool so we thought we'd share it with you.

Note: There are equivalent libraries `https://github.com/cooperhewitt/py-cooperhewitt-unicode` *and an HTTP pony* `https://github.com/cooperhewitt/plumbing-ucd-server` *for* `ucd` *written in Python but they are incomplete compared to the Go version and may eventually be deprecated altogether.*

Comments, suggestions and gentle clue-bats are welcome and encouraged. Enjoy!



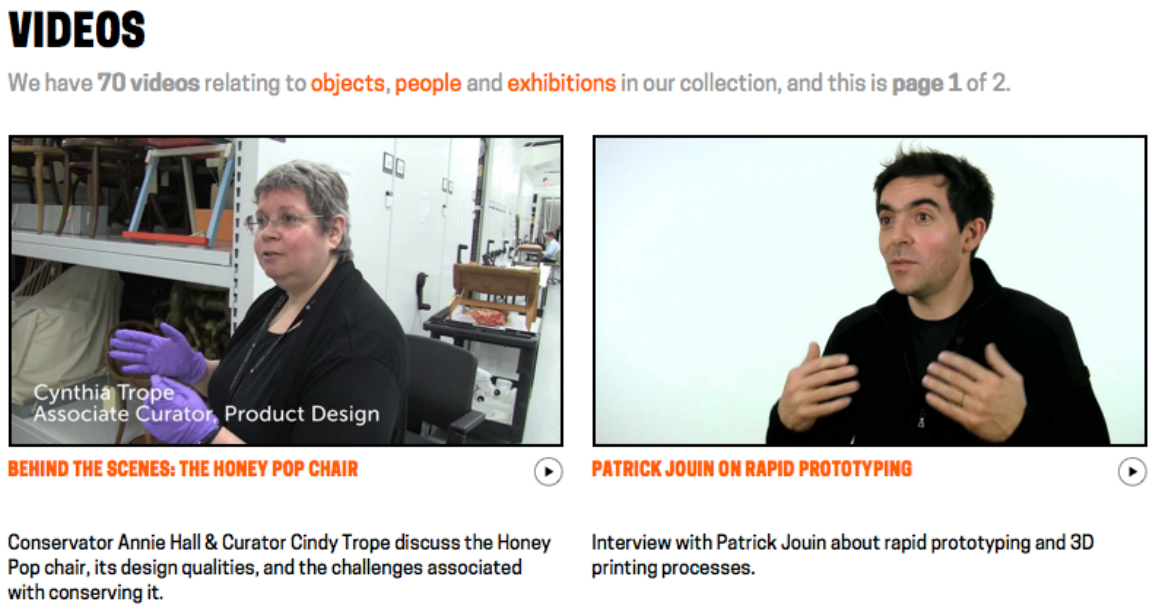
This entry was posted in Backends http://labs.cooperhewitt.org/category/backends/ *on December 2, 2014* [http://labs.cooperhewitt.org/2014/http-ponies/] *by asc* http://labs.cooperhewitt.org/author/asc/.

Sharing our videos, forever

This is one in a series of Labs blogposts exploring the inhouse built technologies and tools that enable everything you see in our galleries.

Our galleries and Pen experience are driven by the idea that everything a visitor can see or do in the museum itself should be accessible later on.

Part of getting the collections site https://collection.cooperhewitt.org and API https://collection.cooperhewitt.org/api (which drives all the interfaces in the galleries designed by Local Projects) ready for re-opening has involved the gathering and, in some cases, generation of data to display with our exhibits and on our new interactive tables. In the coming weeks, I'll be playing blogger catch-up and will write about these new features. Today, I'll start with videos.



https://collection.cooperhewitt.org/videos

Besides the dozens videos produced in-house by Katie http://labs.cooperhewitt.org/author/katieshelly/ – such as the amazing Design Dictionary https://www.youtube.com/playlist?list=PLqwPGOOIhKSDEqOg8OxfciHIMTfyJa9vu series – we have other videos relating to people, objects and exhibitions in the museum. Currently, these are all streamed on our YouTube channel https://www.youtube.com/user/cooperhewitt . While this made hosting much easier, it meant that videos were not easily related to the rest of our

collection and therefore much harder to find. In the past, there were also many videos that we simply didn't have the rights to show after their related exhibition had ended, and all the research and work that went into producing the video was lost to anyone who missed it in the gallery. A large part of this effort was ensuring that we have the rights to keep these videos public, and so we are immensely grateful to Matthew Kennedy, who handles all our image rights, for doing that hard work.

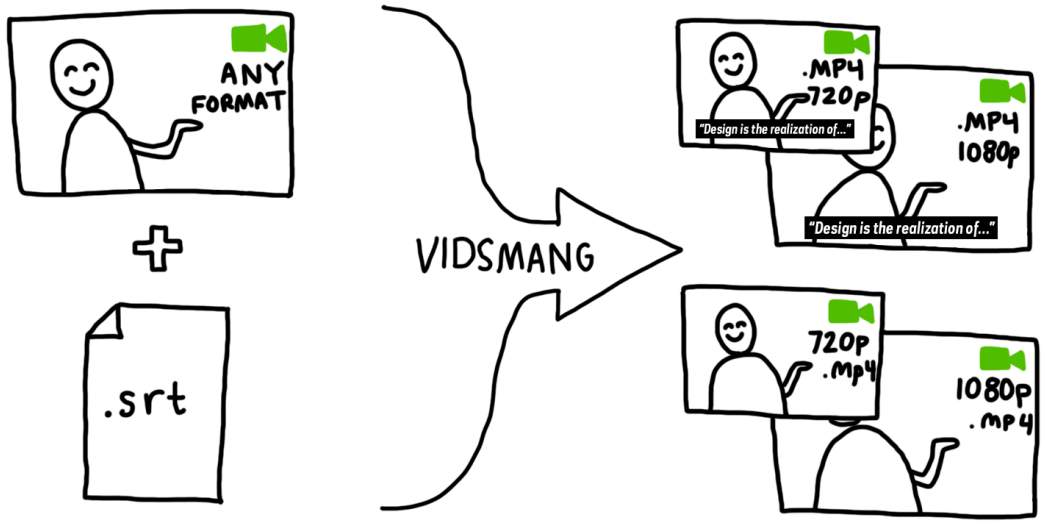
A few months ago, we began the process of adding videos and their metadata in to our collections website and API. As a result, when you take a look at our page for Tokujin Yoshioka's Honey-Pop chair http://collection.cooperhewitt.org/objects/18714653/ , below the object metadata, you can see its related video in which our curators and conservators discuss its unique qualities. Similarly, when you visit our page for our former director, the late Bill Moggridge http://collection.cooperhewitt.org/people/18062553/ , you can see two videos featuring him, which in turn link to their own exhibitions and objects. Or, if you'd prefer, you can just see all of our videos here. https://collection.cooperhewitt.org/videos/

In addition to its inclusion in the website, video data is also now available over our API. When calling an API method for an object https://collection.cooperhewitt.org/api/methods/cooperhewitt.objects.getInfo , person https://collection.cooperhewitt.org/api/methods/cooperhewitt.people.getInfo or exhibition https://collection.cooperhewitt.org/api/methods/cooperhewitt.exhibitions.getInfo from our collection, paths to the various video sizes, formats and subtitle files are returned. Here's an example response for one of Bill's two videos:


```
{
  "id": "68764297",
  "youtube_url": "www.youtube.com\/watch?v=DAHHSS_WgfI",
  "title": "Bill Moggridge on Interaction Design",
  "description": "Bill Moggridge, industrial design-
er and co-founder of IDEO, talks about the advent of interaction design.",
  "formats": {
    "mp4": {
      "1080": "https:\/\/s3.amazonaws.com\/videos.collection.cooperhewit-
t.org\/DIGVID0059_1080.mp4",
      "1080_subtitled": "https:\/\/s3.amazonaws.com\/videos.collection.coop-
erhewitt.org\/DIGVID0059_1080_s.mp4",
      "720": "https:\/\/s3.amazonaws.com\/videos.collection.cooperhewit-
t.org\/DIGVID0059_720.mp4",
```

```
      "720_subtitled": "https:\/\/s3.amazonaws.com\/videos.collection.cooper-
hewitt.org\/DIGVID0059_720_s.mp4"
    }
  },
  "srt": "https:\/\/s3.amazonaws.com\/videos.collection.cooperhewit-
t.org\/DIGVID0059.srt"
}
```

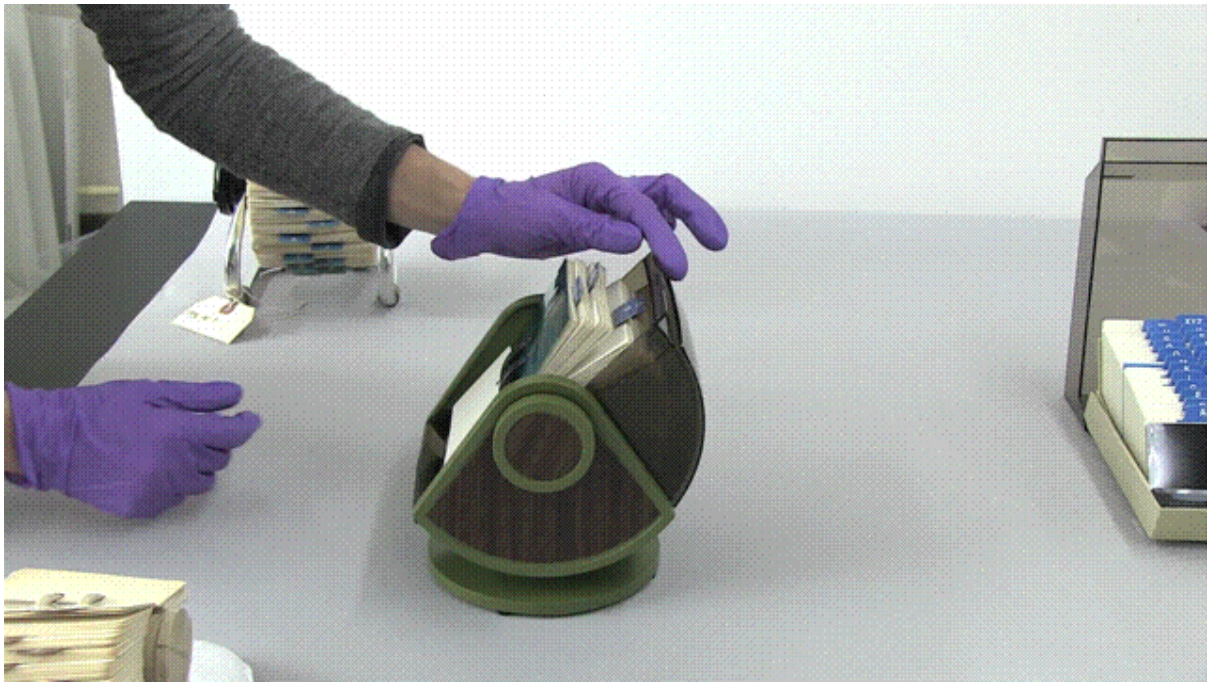

The first step in accomplishing this was to process the videos into all the formats we would need. To facilitate this task, I built VidSmanger https://github.com/cooperhewitt/vidsmanger, which processes source videos of multiple sizes and formats into consistent, predictable derivative versions. At its core, VidSmanger is a wrapper around ffmpeg https://www.ffmpeg.org/, an open-source multimedia encoding program. As its input, VidSmanger takes a folder of source videos and, optionally, a folder of SRT subtitle files. It outputs various sizes (currently 1280×720 and 1920×1080), various formats (currently only mp4, though any ffmpeg-supported codec will work), and will bake-in subtitles for in-gallery display. It gives all of these derivative versions predictable names that we will use when constructing the API response.



http://3ug4ii3uortgp68ee31cybxjyz.wpengine.netdna-cdn.com/wp-content/uploads/2014/12/VidsmangCartoon.png

Because VidSmanger is a shell script composed mostly of simple command line commands, it is easily augmented. We hope to add animated gif generation for our thumbnail images and automatic S3 uploading into the process soon. Here's a proof-of-concept gif generated over the command line using these instructions http://blog.room208.org/post/48793543478. We could easi-

ly add the appropriate commands into VidSmanger so these get made for every video.



http://3ug4ii3uortgp68ee31cybxjyz.wpengine.netdna-cdn.com/wp-content/uploads/2014/12/anim.gif

For now, VidSmanger is open-source and available on our GitHub page! https://github.com/cooperhewitt/vidsmanger To use it, first clone the repo and the run:

```
./bin/init.sh
```

This will initialize the folder structure and install any dependencies (homebrew http://brew.sh/ and ffmpeg). Then add all your videos to the `source-to-encode` folder and run:

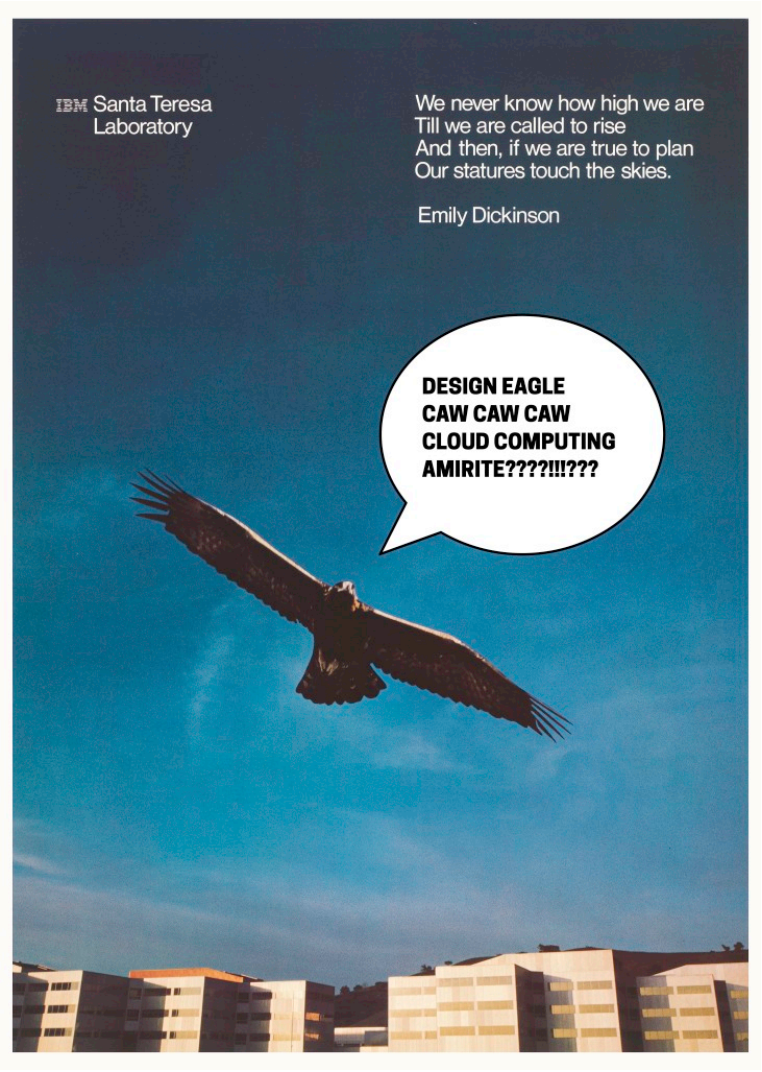
```
./bin/encode.sh
```

Now you're smanging!

This entry was posted in Uncategorized http://labs.cooperhewitt.org/category/uncategorized/ *on December 10,*

2014 [http://labs.cooperhewitt.org/2014/sharing-our-videos-forever/] *by Sam Brenner* http://labs.cooperhewitt.org/author/brenners/.

API methods (new and old) to reflect reality



https://collection.cooperhewitt.org/objects/18618005/

A quick end-of-week blog post to mention that now that the museum has re-opened http://www.newyorker.com/culture/culture-desk/new-cooper-hewitt we have updated the cooperhewitt.galleries.openingHours https://collection.cooperhewitt.org/api/methods/cooperhewitt.galleries.openingHours and cooperhewitt.galleries.isOpen https://collection.cooperhewitt.org/api/methods/cooperhewitt.galleries.isOpen API methods to reflect… well, reality.

In addition to the `cooperhewitt.galleries` API methods we've also published corresponding openingHours https://collection.cooperhewitt.org/api/methods/cooperhewitt.cafe.openingHours/explore/ and isOpen https://collection.cooperhewitt.org/api/methods/cooperhewitt.cafe.isOpen/explore/ methods for the cafe!

For example, cooperhewitt.galleries.isOpen https://collection.cooperhewitt.org/api/methods/cooperhewitt.galleries.isOpen/explore/.

```
curl 'https://api.collection.cooperhewitt.org/rest/?method=cooperhewitt.galleries.isOpen&access_token=***'

{
        "open": 0,
        "holiday": 0,
        "hours": {
                "open": "10:00",
                "close": "18:00"
        },
        "time": "18:01",
        "timezone": "America/New_York",
        "stat": "ok"
}
```

Or, cooperhewitt.cafe.openingHours https://collection.cooperhewitt.org/api/methods/cooperhewitt.cafe.openingHours/explore/.

```
curl -X GET 'https://api.collection.cooperhewitt.org/rest/?method=cooperhewitt.cafe.openingHours&access_token=***'

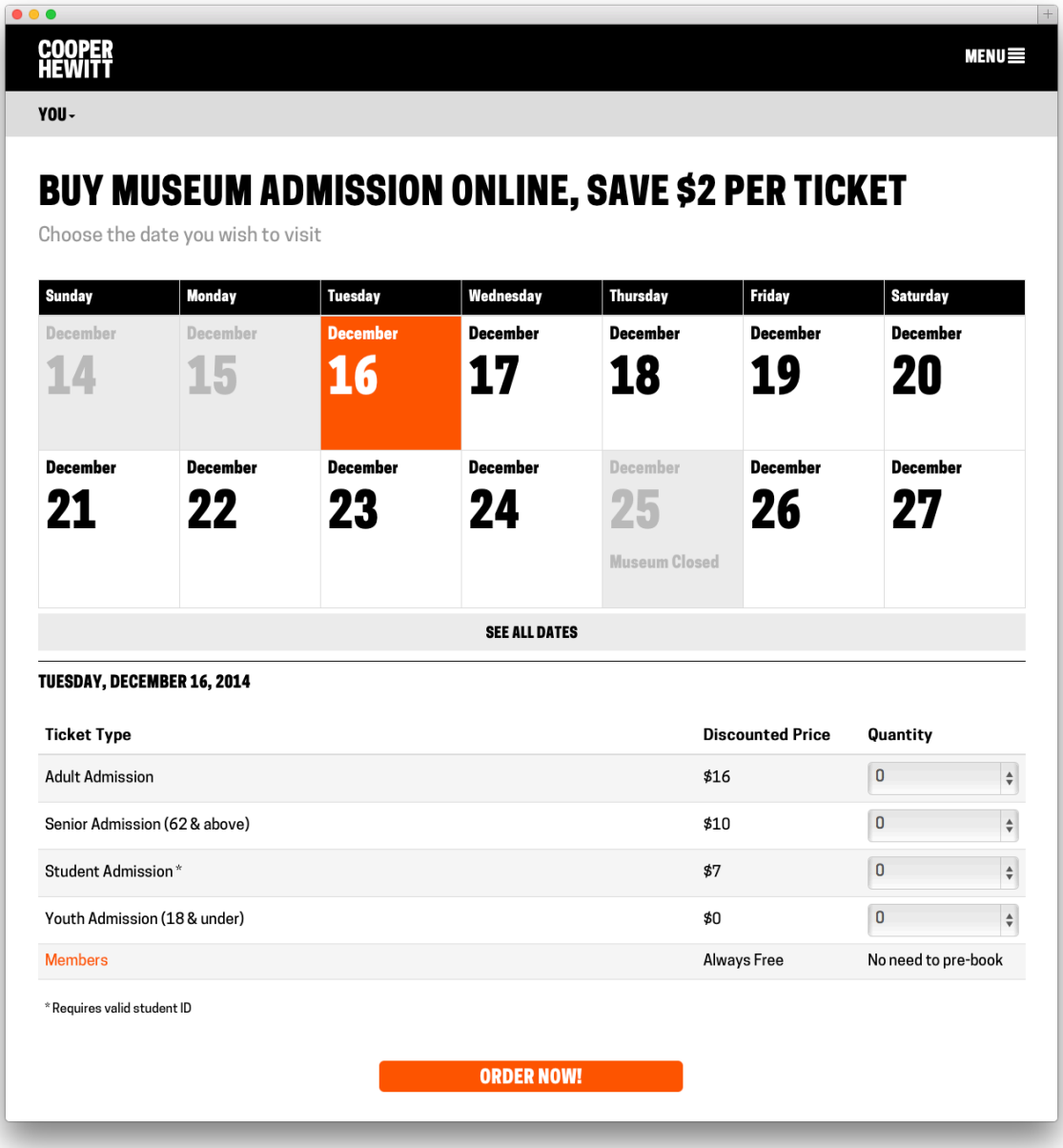
{
        "hours": {
                "Sunday": {
                        "open": "07:30",
                        "close": "18:00"
                },
                "Monday": {
```

```
                        "open": "07:30",
                        "close": "18:00"
                },
                "Tuesday": {
                        "open": "07:30",
                        "close": "18:00"
                },
                "Wednesday": {
                        "open": "07:30",
                        "close": "18:00"
                },
                "Thursday": {
                        "open": "07:30",
                        "close": "18:00"
                },
                "Friday": {
                        "open": "07:30",
                        "close": "18:00"
                },
                "Saturday": {
                        "open": "07:30",
                        "close": "21:00"
                }
        },
        "timezone": "America/New_York",
        "stat": "ok"
}
```


Because coffee http://themuseumofthefuture.com/2012/09/29/what-is-the-best-cultural-venue-to-drink-a-coffee-and-why-we-should-care/, right?

This entry was posted in Backends http://labs.cooperhewitt.org/category/backends/ *on December 12, 2014* [http://labs.cooperhewitt.org/2014/api-methods-to-reflect-reality/] *by asc* http://labs.cooperhewitt.org/author/asc/.

Our new ticketing website



`https://tickets.cooperhewitt.org`

This past week we launched https://tickets.cooperhewitt.org `https://tickets.cooperhewitt.org` — a new online ticketing system which leverages our Consitutent Relationship Management application, Tessitura `http://www.tessituranetwork.com`, as its “source of truth.”

It's a simple application, really. It lets you pick the day you want to come visit us, select the kind of tickets you want to buy, and then you fill out your basic info, plug in your credit card digits and off you go. Moments later you receive an email with PDF versions of your tickets attached.

On the user-facing side of things, it is designed to be as simple as possible. You don't need to log in, there is no "shopping cart", and above all, you can do all of this from your phone if you want to skip ahead of the lines this Winter on your first visit back since we closed nearly three years ago.

A little background

The idea to pre-book tickets online came at us from a number of directions. Some time last year we decided to invest in Tessitura to handle all of our CRM needs. Tessitura, if you have never heard of it before, is an enterprise class, battleship that grew out of the Met Opera House http://www.metopera.org and has made its way around the performing arts sector. It's a great tool if you are looking to centralize everything there is to know about a Constituent. As a museum, it is also appealing in that it does many of the things that non-profit type cultural institutions need to do out of the box.

So, Tessitura. It is now a thing at our museum. Everyone on our staff started ramping up on the software and getting settled into the idea of using Tessitura for one thing or another. Our department began to get requests.

Obviously our membership department would like to use Tessitura to sell and manage memberships. Development would like to use it to manage and collect donations and gifts. Education would like to centralize all public programs, book tours, manage special events, and all of the other crazy things they do. And did I mention we have a museum that sells tickets?

This is how it always starts. The avalanche of ideas, whiteboard sessions, product demos and gentle emails that say things like "When will Tessitura be ready?" begin to pour in. You have to soak it all in and then wring it all out.

The Simplest, Dumbest Thing

Aaron http://twitter.com/thisisaaronland says that quite often. "Just do the simplest, dumbest thing…"

and he's right. Often times you have to boil things down a bit to get to the real core issues at hand. It was clear from the start that this would be an essential part of the "design process" on this project.

So, I started out by asking myself this question: "What is the most basic thing we want to do with Tessitura?"

I wound up with two clear answers.

1. We wanted to be able to sell tickets online. Just basic, general admission tickets. Nothing fancy yet.
2. We eventually want to use Tessitura as our identity provider, and as a way to pair your ticket with the Pen http://www.cooperhewitt.org/new-experience/designing-pen/. More on that towards the end…

Tackling Tickets

So to get started, I thought about the challenge of selling a ticket online. I looked at other sites I liked such as StubHub http://stubhub.com, EventBrite http://eventbrite.com, and other venue websites that I knew used Tessitura like BAM http://www.bam.org, Jazz at Lincoln Center http://www.jazz.org, and the 92Y http://92y.org. I did some research, I bought some tickets, and I asked all my friends who used these sites what they liked and disliked. Eventually I started to find my way gravitating towards the Eventbrite way of doing things. We have been using Eventbrite for a couple of years here at Cooper Hewitt http://labs.cooperhewitt.org/2012/upending-ticketing/, for the most part as a way to sell tickets to education programs and events.

To tell you the truth, Eventbrite has been a dream come true for us and our sales for these events, both paid and free, have been very good. So, what is Eventbrite doing right? Simply put they've made the process of purchasing a ticket to an event online stupidly simple.

I wanted to know more. So, I spent some time and slowly walked myself through the process of booking all kinds of things on Eventbrite. I tried to step through each page in the process, I tried to notice what kind of user feedback I got, and what sort of emails and notifications I got. I tried the same on mobile devices and through their iPhone app. Here are a few takeaways.

1. You don’t need to register in order to purchase a ticket on Eventbrite.
2. If you don’t register when making an initial purchase, you can register later and see your purchase history.
3. As soon as you book your tickets, you get them in an email.
4. The Thank You page is just as useful when you are logged out as when you are logged in.
5. Most importantly, you can only buy one thing at a time. In other words, there is no idea of a Shopping Cart.

That last one was pretty huge. Most eCommerce sites are built around the idea that users put items in a cart and then “Checkout.” Eventbrite doesn’t do it this way. Instead, you simply pick the thing you are wanting to attend, select the kinds of tickets you want (student, senior, etc) and then put in your credit card info. Once you hit submit, you’ve paid for your tickets and your transaction is complete.

I felt this flow was incredibly powerful and probably one of the reasons Eventbrite was working so well for our education programs. There are simply less chances to change your mind, less confusion over what you are buying, and the end-to-end process of picking something out and paying for it is just so much smaller than the more traditional shopping cart experience.

I began to think of it kind of like the difference between getting your weekly groceries and just picking up a six pack. The behaviors are totally different because you are trying to accomplish two totally different tasks. One is very routine, requires a little creativity and some patience, and a willingness to wander around and “pick.” The other is a strategic strike, designed to get in and get out so you can get home and relax with a nice cold one.

The Eventbrite concept seemed like what we wanted. I had my simplest dumbest thing, and something to model it on.

Technical Challenges

With every new project comes some kind of technical challenge(s). Tessitura is a “new to us” application and our staff at Cooper Hewitt were clearly at the bottom left of a steep learning curve when we started the project. We also had many challenges we knew we were going to have to face because we are a “Governmental Institution.” So things like PCI compliance, complex network configurations, and security scans were all things I was going to need to

learn about.

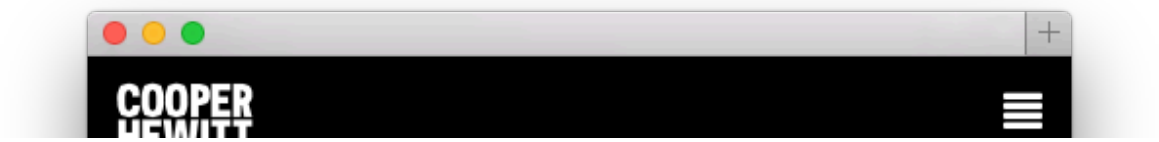
Tessitura comes with two APIs. One is a somewhat older (as in the first thing they built) SOAP API http://en.wikipedia.org/wiki/SOAP , and the other is a newer (as in still under development) REST API http://en.wikipedia.org/wiki/Representational_state_transfer . Both allow data to get in and out of Tessitura in a variety of forms.

In addition to the standard SOAP and REST APIs, Tessitura has the facility to expose just about anything you can build into an MS SQL Stored Procedure http://en.wikipedia.org/wiki/Stored_procedure through its API. This is an incredibly powerful feature, which can also be quite dangerous if you think about it.

When I attended the Tessitura Learning Conference & Convention this past summer, it became clear to me that many institutions that use Tessitura are building some kind of API wrapper, or some type of middleware that helps them make sense of it all. We chose to do the same. To accomplish this, I chose to model the API wrapper off our own Collections API https://collection.cooperhewitt.org/api/ , which is a REST-ish API https://github.com/cooperhewitt/flamework-api based on Flamework http://github.com/exflickr/flamework , and uses oAuth2 https://collection.cooperhewitt.org/api/oauth2/ for authentication. Having this API wrapper allows us to all speak the same language and use the same interface. It is also very, very similar to the Collections API, so among our own staff, it is pretty easy to navigate. The API wrapper, wraps methods from both the Tessitura SOAP and the REST APIs and presents a unified interface to both of them. It doesn't implement every single API method, and it exposes "new" methods that we have custom built via those Stored Procedures I mentioned.

The Tickets website is a separate project that talks directly to the API. It is also a Flamework project, written primarily in PHP. It uses a MySQL database to store a small amount of local data, but for the most part it is making calls to the API wrapper, which in turn is making calls to either the SOAP or REST Tessitura APIs. Tessitura is the source of truth for most of the things the Ticketing website does.

The Front End



BUY MUSEUM ADMISSION ONLINE, SAVE $2 PER TICKET

Choose the date you wish to visit

December 16 **Tuesday**

TUESDAY, DECEMBER 16, 2014

Adult Admission

$16

0

Senior Admission (62 & above)

$10

0

Student Admission *

$7

0

Youth Admission (18 & under)



http://3ug4ii3uortgp68ee31cybxjyz.wpengine.netdna-cdn.com/wp-content/uploads/2014/12/Screenshot-2014-12-16-16.25.44.png

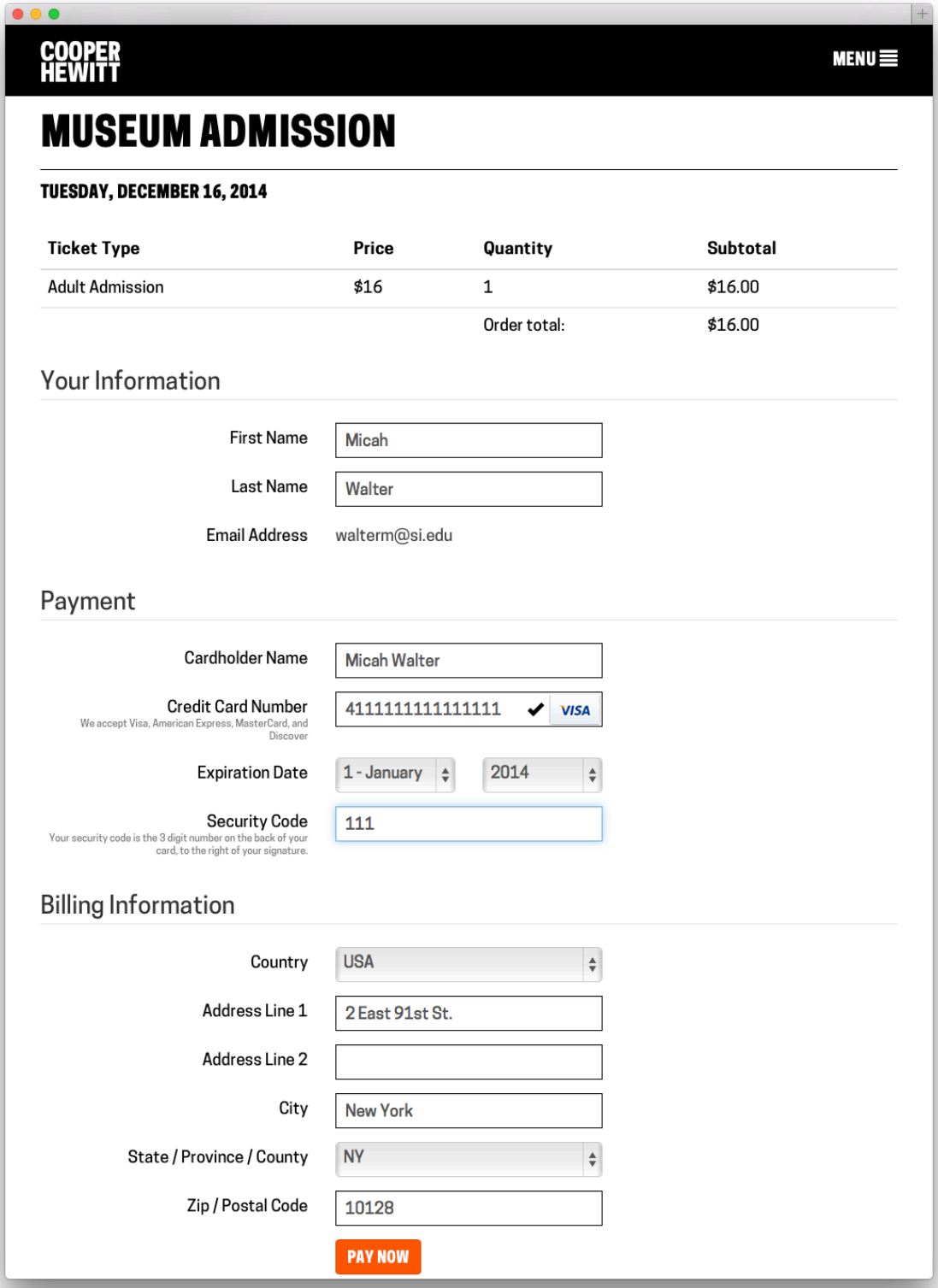
The Tickets site from the user's perspective is designed to be extremely simple. I worked with Sam https://samjbrenner.com (our in house front end guru) to build a responsive, and simple web application that does basically one thing, but the devil is always in the details.

At first glance, all the site does is allow you to select the day you want to come visit us, pick out what kinds of tickets you want, and then fill out your billing info and receive your tickets. It's basically a calendar and a form and not much more. But like I said, the devil is in the details.

First, Sam built a beautiful calendar like one I'd never seen before. We talked at length about how dumb most website calendars were, and we tried to push things in a new direction. Our calendar starts out by showing you what you most likely are looking for–Today. It displays the next bunch of days up to two weeks worth by default, and if you are looking for a special date, it lets you drop down and navigate around until you find it. On mobile its slightly different in that it doesn't show you any past dates (why would you want to book the past?) and it limits things a little so you're not as overwhelmed by the interface. We call this "designing for context" and we thought that users might be using their mobile phones to buy a ticket online and jump up to the front of the queue.

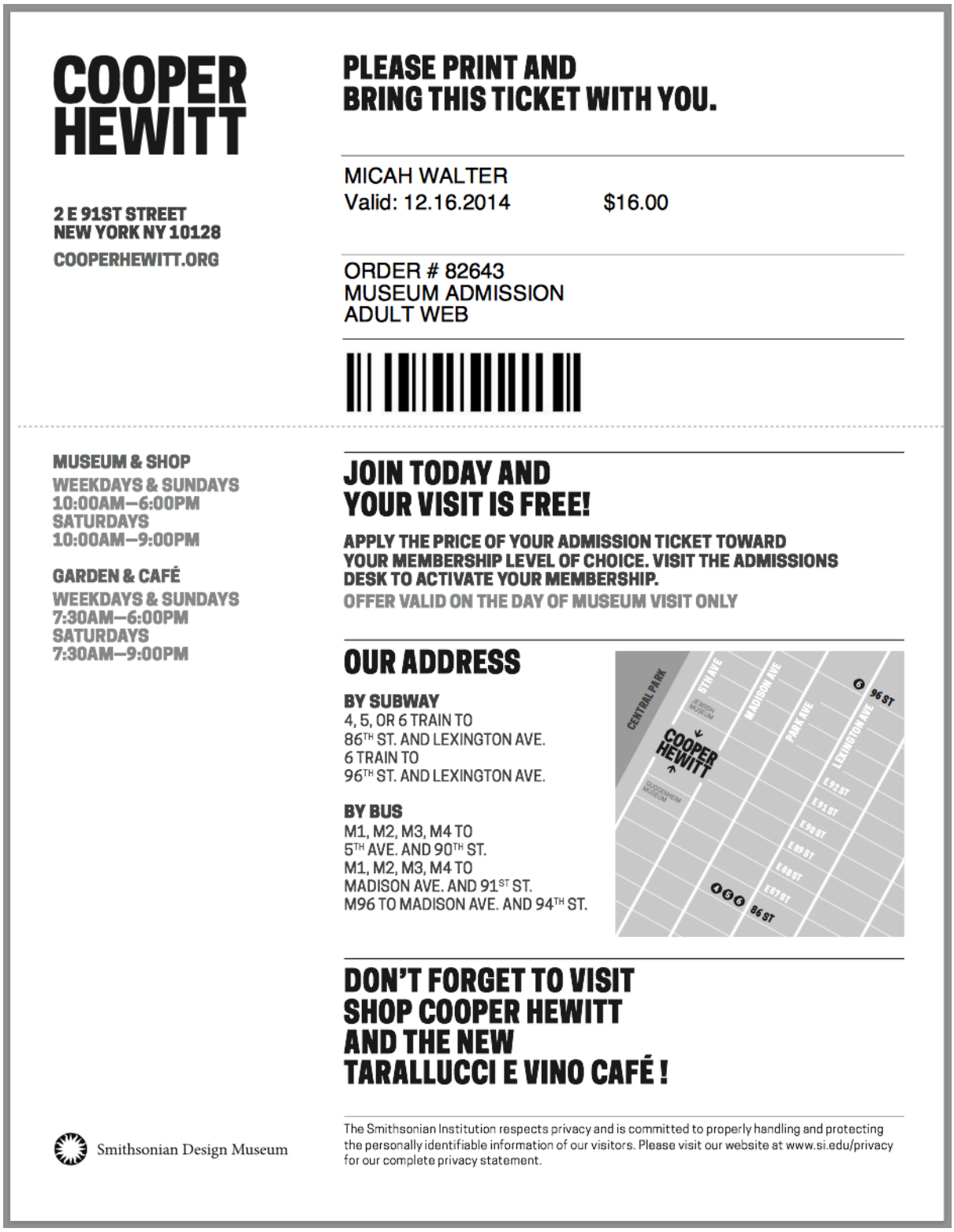
Once you've selected the date you want, the app loads up the available tickets right below the calendar. You can easily change your mind and pick a different date. From here you just select the type and quantity of each ticket you want. Sam's code does a bunch of front-end validations to make sure everything you are trying to do makes sense (you can only purchase a youth ticket with another paid ticket for example). Between the two of us, we try to do as much validation to what you are selecting as possible, both in Javascript on the front-end and in PHP on the back.

Once you hit Order Now, an order form is generated and displayed. I think its important to note here that nothing has really happened yet in Tessitura-land. We asked Tessitura for some details about the tickets you are interested in, but we haven't "added them to your cart" or anything like that as of yet.



http://3ug4ii3uortgp68ee31cybxjyz.wpengine.netdna-cdn.com/wp-content/uploads/2014/12/Screenshot-2014-12-16-16.28.34.png

You can then fill out your vitals. We ask you to give us your name, email, credit card details and billing address. We store all of this, with the exception of your credit card, in Tessitura. We make you an account, and at this point we send you an activation email which allows you to set up your password at your leisure. If all goes well with your credit card, we build your tickets (I chose to do all this with FPDF http://www.fpdf.org rather than try and use Tessitura's built in Print at Home server thing) send them in an email, and then take you to a Thank You Page. You never had to register, or log in, and you technically never do. Your PDF tickets arrive in your inbox and that is technically all you need.

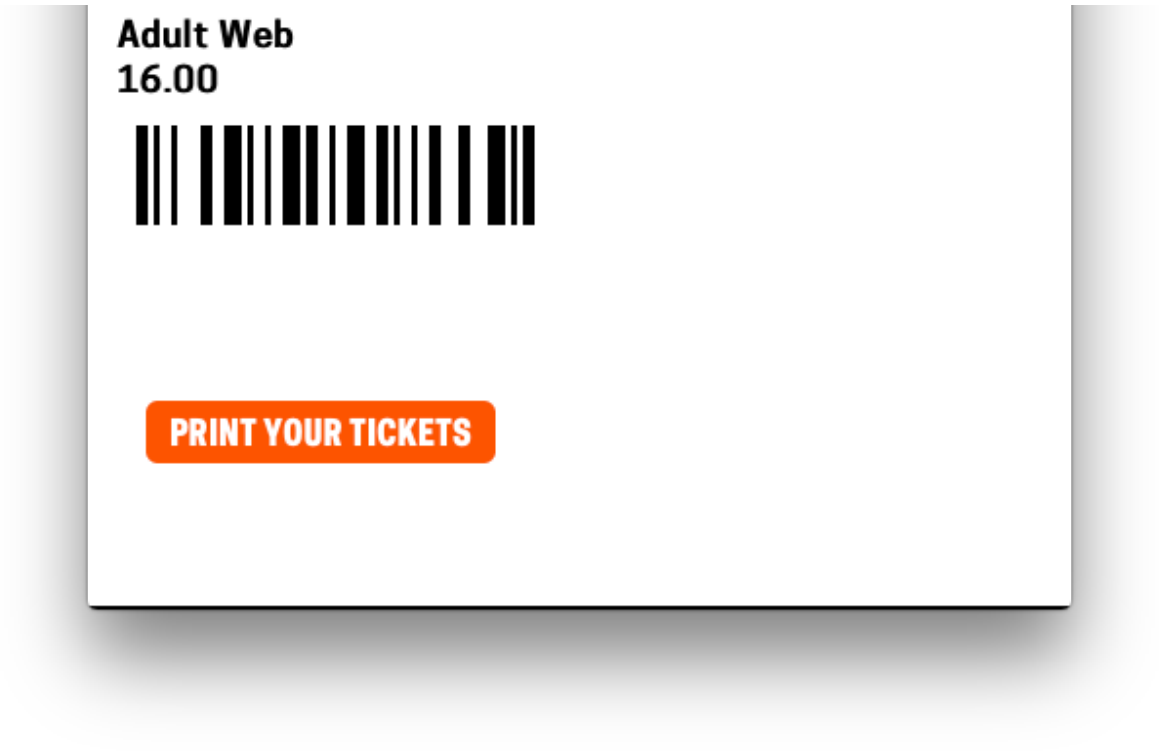


http://3ug4ii3uortgp68ee31cybxjyz.wpengine.netdna-cdn.com/wp-content/uploads/2014/12/Screenshot-2014-12-16-16.27.35.png

As a little bonus, we just stick the barcode and some basic metadata about each ticket you

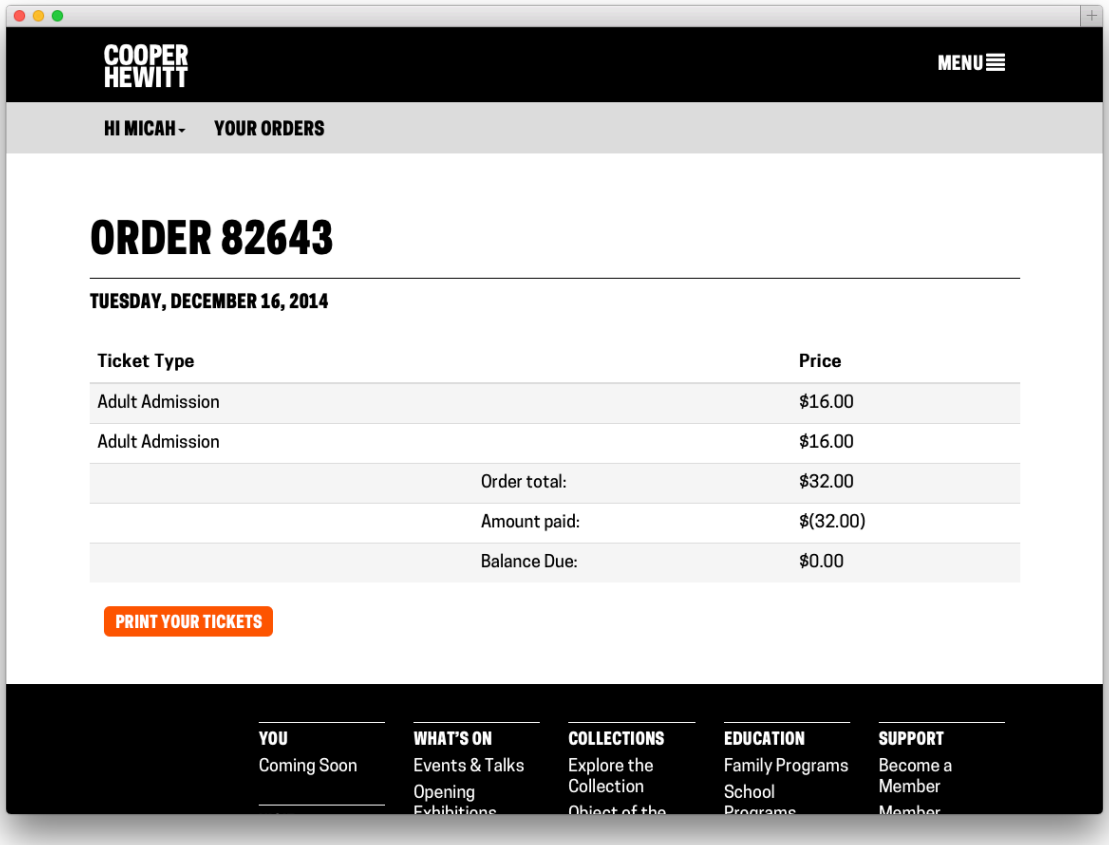
bought on the Thank You Page so you can just present your phone at the door. This part is still a little rough and I chose to leave it that way for the time being so we can do some user research in the galleries. It's a nice feature, but only time will tell if people actually use it or if it needs some finesse.





http://3ug4ii3uortgp68ee31cybxjyz.wpengine.netdna-cdn.com/wp-content/uploads/2014/12/Screenshot-2014-12-16-16.26.50.png

Now that you are in the system, you can buy more tickets using the same email address and they will be connected to your same account, even if you still have never logged in. If you do choose to activate your account and login, you can look at your order history, and reprint your tickets if you've lost your email copy.



http://3ug4ii3uortgp68ee31cybxjyz.wpengine.netdna-cdn.com/wp-content/uploads/2014/12/Screenshot-2014-12-16-16.27.10.png

Tessitura & The Pen

A while back in this blog post, I mentioned that we also wanted to use Tessitura as our identity provider and as a way to pair the ticket you've purchased with the pen we've handed you. This work is nearly done, but not yet in production. It will go live when our pen is available sometime in early 2015. But, the short story is, when you buy a ticket, either online or in person, we generate a special coded version of your ticket. This code gets paired up with the internal ID of the pen we gave you and that pairing gets stored in a database. What this all amounts to is that when you get home, and you want to see all the cool things you've done with your pen, you simply enter the code (or go to a custom short URL) on our website. We look up your pairing and are able to connect your Identity (Tessitura) with your Visit (on the Collections site). But that is all the topic of a future series of blog posts.

Next

Now that the Tickets site is up and running, and we are watching the sales roll in, it's easy to start thinking of more features and new ways to expand what the site can do. I've already started building simple admin tools and have been thinking about building a basic check-in app for off-site events. It's too early to talk about all of the things we aim to do and how we plan to expand our online sales, but I'm hopeful that we will stay focused and narrow in our approach, offering our users the most elegant visitor experience possible. Or at the very least, the simplest, dumbest thing.

This entry was posted in Backends http://labs.cooperhewitt.org/category/backends/, *Ticketing* http://labs.cooperhewitt.org/category/ticketing-2/, *UX* http://labs.cooperhewitt.org/category/ux/ *on December 16, 2014* [http://labs.cooperhewitt.org/2014/our-new-ticketing-website/] *by micah* http://labs.cooperhewitt.org/author/micah/.

emacs Cheat Sheet

Due to a frequent need to work off of different servers, I found it necessary to graduate from nano https://en.wikipedia.org/wiki/GNU_nano and up my command line text editor skills. Enter emacs https://en.wikipedia.org/wiki/Emacs! Aaron gave me a quick crash course, from which I generated a cheat sheet of everyday commands to tape to my monitor. Rule #1 of emacs (for me at least) was "forget every keyboard shortcut you've ever known," so having a cheat sheet to remind me that "copy" is "escape key, w key" was necessary until my muscle memory kicked in.

If you're in this situation maybe this cheat sheet will help you too.

Gist is here https://gist.github.com/sambrenner/316eba0eb89e6a5691fe.

```
EMACS CHEAT SHEET

C-g . . . . . . . Stop bothering me
C-x C-c . . . . . Exit Emacs

C-x C-f . . . . . Find File
C-x k . . . . . . Kill Buffer
C-x b . . . . . . Load Buffer
C-x o . . . . . . Next Buffer
C-x left/right  . Next/Previous buffer
C-x [0-3] . . . . Fiddle with buffer views

M-g g . . . . . . Goto Line
C-a . . . . . . . Beginning of line
C-e . . . . . . . End of line
C-v . . . . . . . Page down
M-v . . . . . . . Page up
C-s . . . . . . . Search in buffer
C-x C-s . . . . . Save buffer

C-space . . . . . Set mark
C-w . . . . . . . Cut
M-w . . . . . . . Copy
```

```
C-y . . . . . . . Paste

M-x things:
M-x shell . . . . . . Open Shell
M-p . . . . . . . . . Previous shell command
M-x replace-string  . Find/Replace in file
M-x rgrep . . . . . . Find in folders
M-x list-packages . . Package Manager

Magit:
s . . . . Stage
u . . . . Unstage
c . . . . Commit
k . . . . Discard modification
P . . . . Push
F . . . . Pull
C-c C-c . Save commit message

Dired Mode:
m . . . Mark file
u . . . Unmark file
! . . . Perform shell command on file(s)
```

This entry was posted in Basics http://labs.cooperhewitt.org/category/basics/ *and tagged cheat sheet* http://labs.cooperhewitt.org/tag/cheat-sheet/, *dev tools* http://labs.cooperhewitt.org/tag/dev-tools/, *emacs* http://labs.cooperhewitt.org/tag/emacs/, *text editor* http://labs.cooperhewitt.org/tag/text-editor/ *on December 18, 2014* [http://labs.cooperhewitt.org/2014/emacs-cheat-sheet/] *by Sam Brenner* http://labs.cooperhewitt.org/author/brenners/.